aws

Guía para desarrolladores

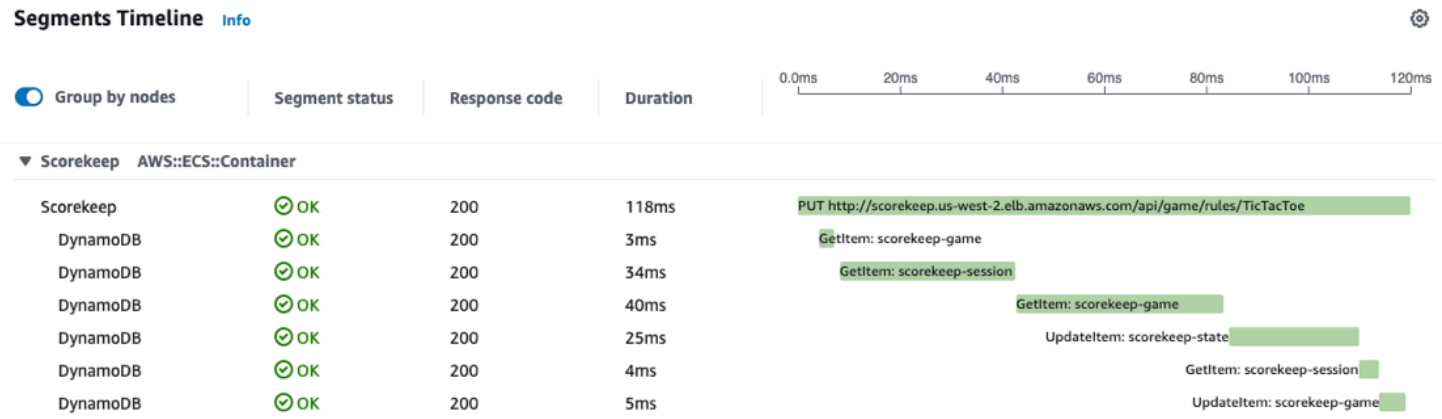
AWS X-Ray

AWS X-Ray: Guía para desarrolladores

Table of Contents

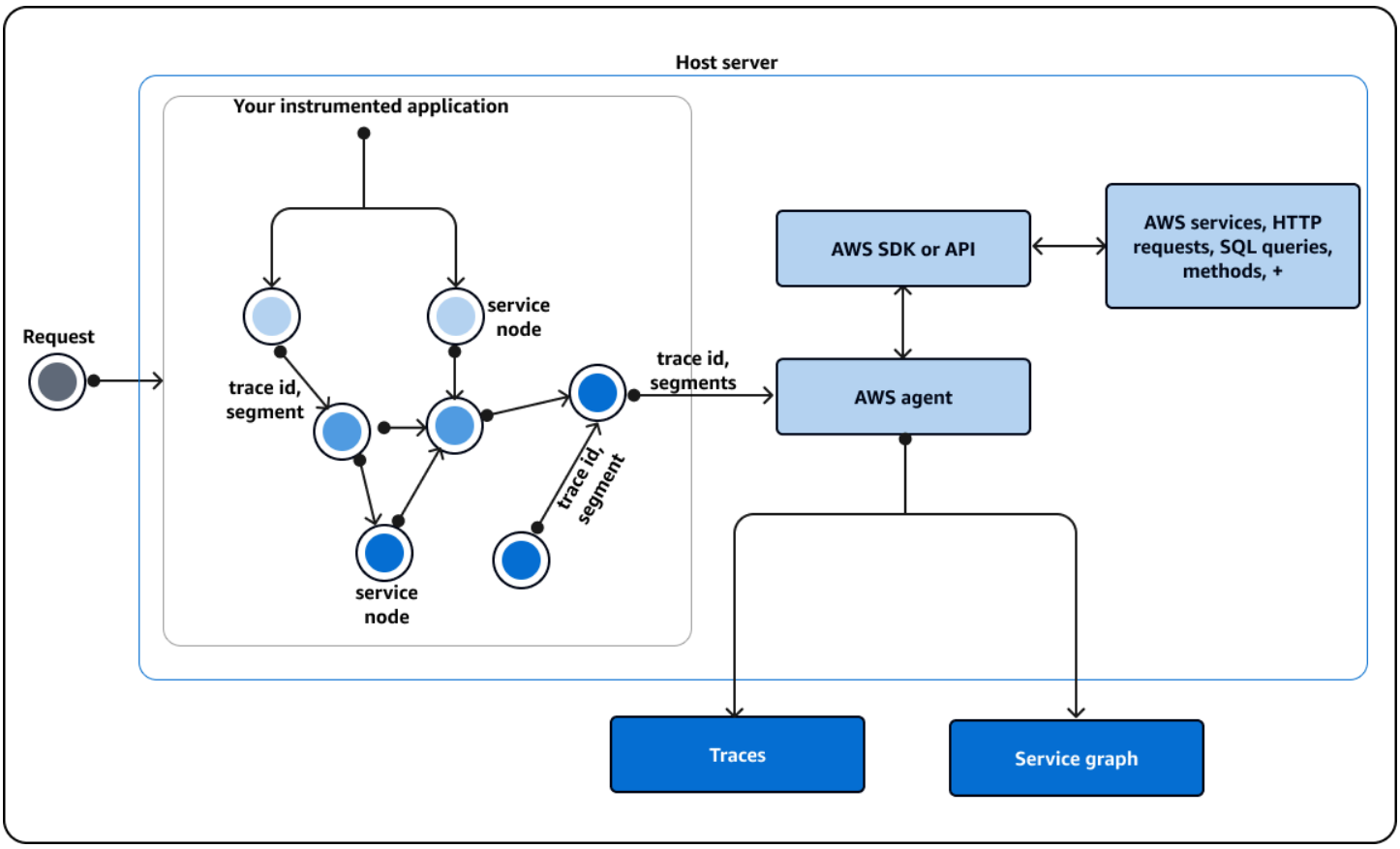
¿Qué es () AWS X-Ray?

AWS X-Ray es un servicio que recopila datos sobre las solicitudes que su aplicación atiende y proporciona herramientas que puede utilizar para consultar, filtrar y obtener información sobre dichos datos para identificar problemas y oportunidades de optimización. En cada solicitud rastreada enviada a su aplicación, puede ver información detallada sobre la solicitud, la respuesta y también sobre las llamadas que su aplicación realiza a recursos, microservicios, bases de datos de AWS y API web posteriores.



AWS X-Ray recibe rastros de la aplicación del usuario, además de los Servicios de AWS utilizados por su aplicación que ya están integrados en X-Ray. La instrumentación de una aplicación implica el envío de datos de rastro para solicitudes entrantes y salientes y otros eventos de la aplicación junto con los metadatos de cada solicitud. Muchos escenarios de instrumentación solo requieren cambios en la configuración. Por ejemplo, puede instrumentar todas las solicitudes HTTP entrantes y las llamadas posteriores a los Servicios de AWS que haga su aplicación de Java. Existen varios SDK, agentes y herramientas que puede utilizar para instrumentar su aplicación para rastreo de X-Ray. Para obtener más información, consulte Instrumentación de su aplicación.

Los Servicios de AWS que están integrados en X-Ray pueden añadir encabezados de rastreo a las solicitudes entrantes, enviar datos de rastro a X-Ray o ejecutar el daemon X-Ray. Por ejemplo, AWS Lambda puede enviar datos de rastro sobre solicitudes a las funciones de Lambda del usuario y ejecutar el daemon de X-Ray en procesos de trabajo para facilitar el uso del SDK de X-Ray.



En lugar de enviar los datos de rastro directamente a X-Ray, cada SDK cliente envía documentos de segmento JSON a un proceso del daemon que escucha el tráfico UDP. El daemon de X-Ray almacena en búfer segmentos en una cola y los carga en X-Ray en lotes. El demonio está disponible para Linux, Windows y macOS y se incluye en las plataformas AWS Elastic Beanstalk y AWS Lambda.

X-Ray utiliza datos de rastro de los recursos de AWS que alimentan las aplicaciones del usuario en la nube para generar un mapa de rastros detallado. El mapa de rastros muestra el cliente, el servicio frontend y los servicios backend a los que llama dicho servicio frontend para procesar solicitudes y mantener los datos. Utilice el mapa de rastros para identificar cuellos de botella, picos de latencia y otros problemas que puede resolver a fin de mejorar el desempeño de las aplicaciones.


scorekeep-move
DynamoDB Table
Client
Scorekeep
ECS Container
SNS
SNS
scorekeep-session
DynamoDB Table

Introducción a X-Ray

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Para usar X-Ray, siga estos pasos:

1. Instrumente su aplicación de manera que X-Ray pueda realizar un rastreo de cómo su aplicación procesa una solicitud.

   - Utilice las señales X-Ray SDKs APIs, X-Ray ADOT o CloudWatch Application para enviar datos de rastreo a X-Ray. Para obtener más información acerca de qué interfaz utilizar, consulte Elección de una interfaz.

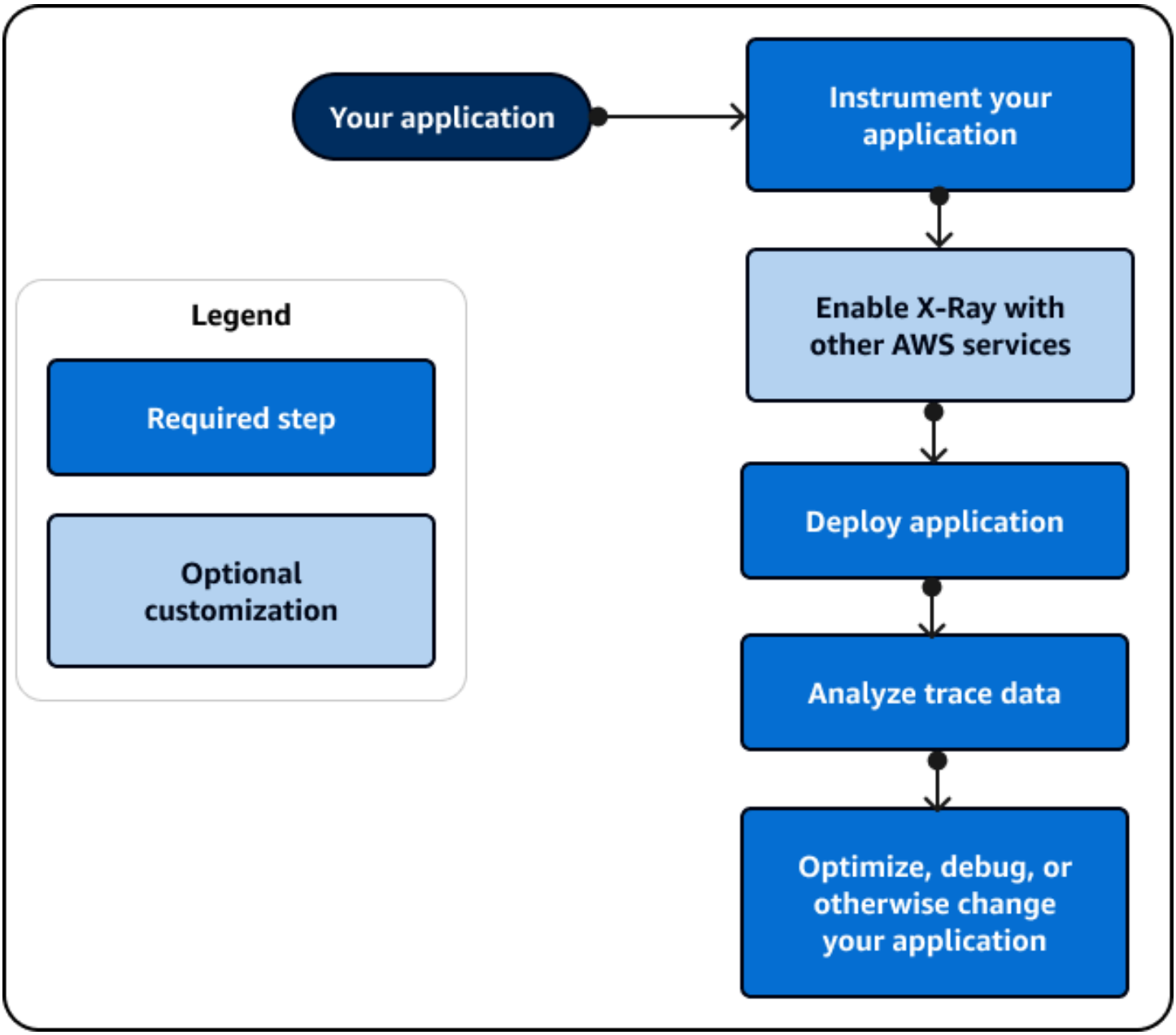
   Para obtener más información acerca de la instrumentación, consulte Instrumentación de su solicitud para AWS X-Ray.
2. (Opcional) Configure X-Ray para que funcione con otros Servicios de AWS que se integren con X-Ray. Puede muestrear los rastros y añadir encabezados a las solicitudes entrantes, ejecutar un agente o recopilador y enviar automáticamente los datos de rastro a X-Ray. Para obtener más información, consulte Integración AWS X-Ray con otros Servicios de AWS.
3. Implemente la aplicación instrumentada. A medida que su aplicación reciba solicitudes, el SDK de X-Ray registrará los datos de rastro, segmentos y subsegmentos. En este paso también es posible que deba configurar una política de IAM e implementar un agente o recopilador.

   - Para ver ejemplos de scripts para implementar una aplicación mediante el SDK AWS Distro for OpenTelemetry (ADOT) y el CloudWatch agente en distintas plataformas, consulte los scripts de demostración de Application Signals.

- Para ver un script de ejemplo para implementar una aplicación mediante el SDK de X-Ray y el daemon de X-Ray, consulte AWS X-Ray ejemplo de aplicación.

4. (Opcional) Abra una consola para ver y analizar los datos. Para inspeccionar el funcionamiento de su aplicación, es posible consultar una representación gráfica de un mapa de rastros, un mapa de servicio u otros. Utilice la información gráfica de la consola para optimizar, depurar y comprender su aplicación. Para obtener más información sobre cómo elegir una consola, consulte Uso de una consola.

En el siguiente diagrama se muestra cómo empezar a utilizar X-Ray:



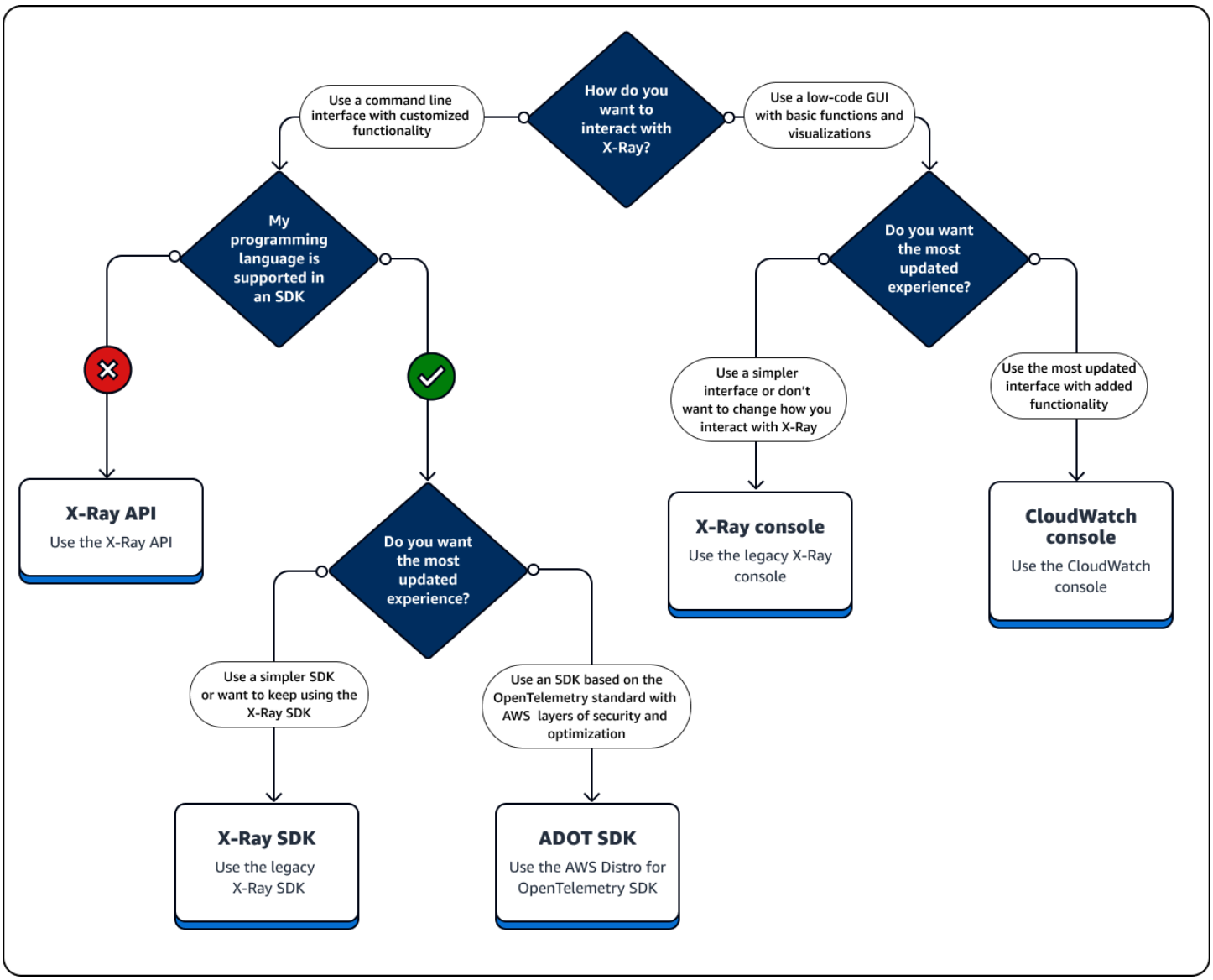
Para ver un ejemplo de los datos y mapas disponibles en la consola, inicie una aplicación de ejemplo que ya esté instrumentada para generar datos de rastro. En unos minutos, podrá generar tráfico, enviar segmentos a X-Ray y ver un mapa de servicio y rastro.

Elección de una interfaz

AWS X-Ray puede proporcionar información sobre cómo funciona su aplicación y qué tan bien interactúa con otros servicios y recursos. Después de instrumentar o configurar la aplicación, X-Ray recopila los datos de rastro a medida que la aplicación atiende solicitudes. Puede analizar estos datos de rastro para identificar problemas de rendimiento, solucionar errores y optimizar sus recursos. Esta guía le indica cómo interactuar con X-Ray siguiendo las siguientes pautas:

- Utilice una Consola de administración de AWS si quiere empezar rápidamente o si puede utilizar visualizaciones prediseñadas para realizar tareas básicas.
  - Elija la CloudWatch consola Amazon para disfrutar de una experiencia de usuario más actualizada que contenga todas las funciones de la consola X-Ray.
  - Utilice la consola de X-Ray si desea una interfaz más sencilla o no quiere cambiar la forma en que interactúa con X-Ray.
- Usa un SDK si necesitas más funciones de rastreo, monitoreo o registro personalizadas de las que te Consola de administración de AWS puede ofrecer.
  - Elija el SDK de ADOT si quiere un SDK independiente del proveedor y basado en el SDK de OpenTelemetry de código abierto con capas adicionales de optimización y seguridad de AWS .
  - Elija el SDK de X-Ray si prefiere un SDK más sencillo o no quiere actualizar el código de su aplicación.
- Use las operaciones de la API de X-Ray si un SDK no es compatible con el lenguaje de programación de su aplicación.

El siguiente diagrama le ayudará a elegir cómo quiere interactuar con X-Ray:



Explore los distintos tipos de interfaz

- Uso de un SDK
- Uso de una consola
- Uso de la API de X-Ray

Uso de un SDK

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Utilice un SDK si desea utilizar una interfaz de línea de comandos o si necesita más funciones personalizadas de rastreo, supervisión o registro de las que están disponibles en una Consola de administración de AWS. También puede usar un AWS SDK para desarrollar programas que usen el X-Ray APIs. Puede usar el SDK AWS Distro for OpenTelemetry (ADOT) o el SDK X-Ray.

Si utiliza un SDK, puede añadir personalizaciones a su flujo de trabajo tanto al instrumentar su aplicación como al configurar su recopilador o agente. Utilice un SDK para efectuar las siguientes tareas que no puede llevar a cabo con una Consola de administración de AWS:

- Publicar métricas personalizadas: tome muestras de métricas en resoluciones altas de hasta 1 segundo, use varias dimensiones para agregar información sobre una métrica y agregue puntos de datos en un conjunto de estadísticas.
- Personalizar su recopilador: personalice la configuración de cualquier parte de un recopilador, incluidos el receptor, el procesador, el exportador y el conector.
- Personalizar su instrumentación: personalice segmentos y subsegmentos, añada pares clave-valor personalizados como atributos y cree métricas personalizadas.
- Crear y actualizar las reglas de muestreo mediante programación.

Use el ADOT SDK si desea la flexibilidad de usar un OpenTelemetry SDK estandarizado con niveles adicionales de AWS seguridad y optimización. El SDK AWS Distro for OpenTelemetry (ADOT) es un paquete independiente del proveedor que permite la integración con back-ends de otros proveedores y no relacionados con los AWS servicios sin tener que reorganizar el código.

Utilice el SDK de X-Ray si ya lo hace y solo se integra con backends de AWS y no pretenda cambiar la forma en que interactúa con X-Ray o con el código de su aplicación.

Para obtener más información acerca de cada una de las características, consulte Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs.

Uso del SDK de ADOT

El ADOT SDK es un conjunto de bibliotecas y agentes de código abierto que envían APIs datos a los servicios de backend. ADOTes compatible con varios backends y agentes AWS, se integra con ellos y proporciona un gran número de bibliotecas de código abierto mantenidas por la OpenTelemetry comunidad. Utilice el SDK de ADOT para instrumentar su aplicación y recopilar registros, metadatos, métricas y rastros. También puedes usarlo ADOT para monitorear los servicios y configurar una alarma en función de tus métricas. CloudWatch

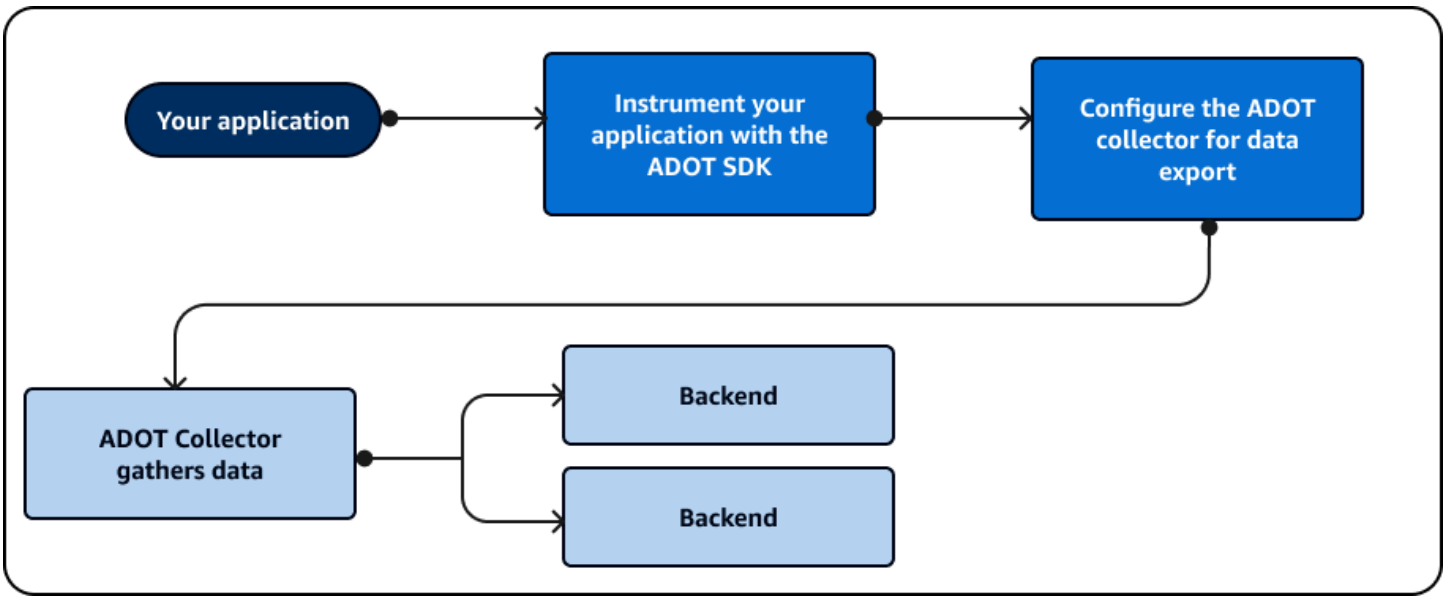
Si utiliza el SDK de ADOT, dispone de las siguientes opciones, en combinación con un agente:

- Usa el ADOT SDK con el CloudWatch agente (se recomienda).
- Utilice el ADOT SDK con el ADOTrecopilador, lo que se recomienda si desea utilizar software independiente del proveedor con AWS niveles de seguridad y optimización.

Para utilizar el SDK de ADOT, haga lo siguiente:

- Instrumente su aplicación con el SDK de ADOT. Para obtener más información, consulte la documentación de su lenguaje de programación en la documentación técnica de ADOT.
- Configure un recopilador de ADOT para que le indique dónde enviar los datos que recopila.

Una vez que el ADOT recopilador recibe los datos, los envía al servidor que especifiques en la ADOT configuración. ADOTpuede enviar datos a varios backends, incluso a proveedores externos AWS, como se muestra en el siguiente diagrama:



AWS se actualiza periódicamente ADOT para añadir funcionalidad y alinearse con el OpenTelemetrymarco. Las actualizaciones y los planes futuros para el desarrollo de ADOT forman parte de una hoja de ruta que está disponible públicamente. ADOT admite varios lenguajes de programación, entre los que se incluyen los siguientes:

- Go
- Java
- JavaScript
- Python
- .NET
- Ruby
- PHP

Si está utilizando Python, ADOT puede instrumentar automáticamente su aplicación. Para empezar a ADOT utilizarla, consulte Introducción y introducción a la AWS distribución de OpenTelemetry Collector.

Uso del SDK de X-Ray

El SDK de X-Ray es un conjunto de AWS APIs bibliotecas que envían datos a los servicios de AWS backend. Utilice el SDK de X-Ray con el fin de instrumentar su aplicación y recopilar datos de rastro. No puede utilizar el SDK de X-Ray para recopilar datos de registro o de métricas.

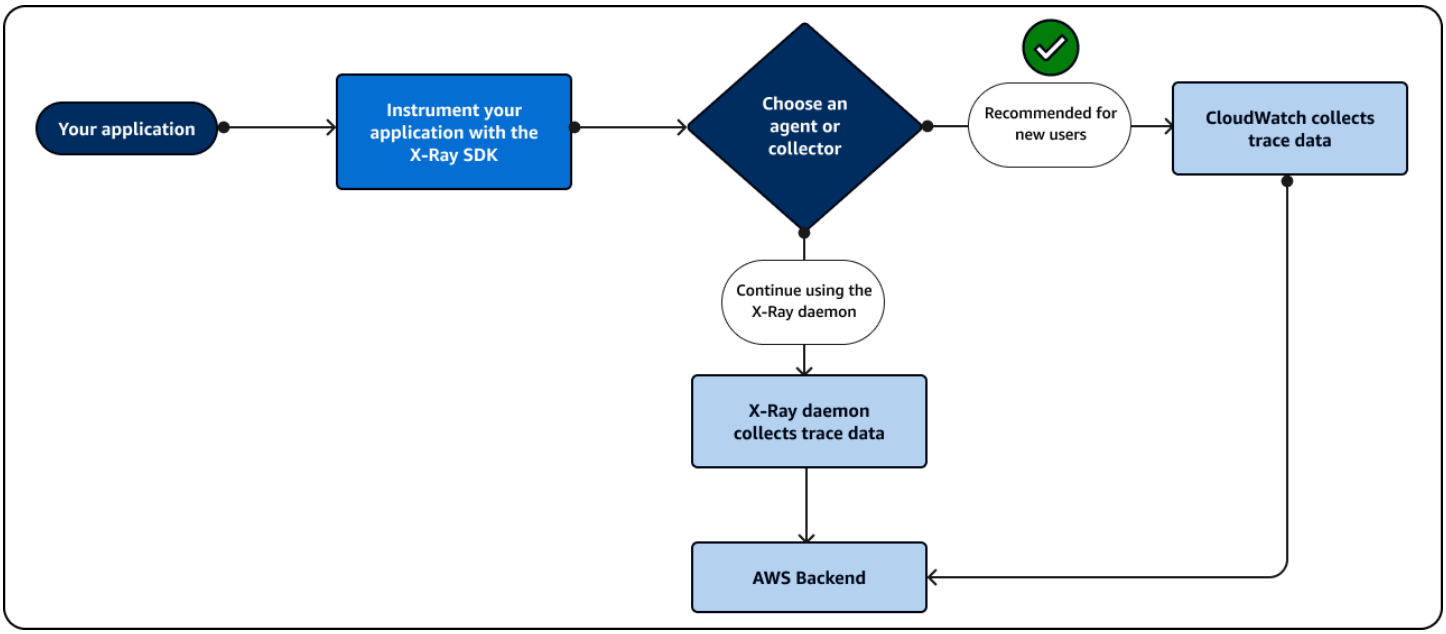
Si utiliza el SDK de X-Ray, dispone de las siguientes opciones, en combinación con un agente:

- Uso del SDK de X-Ray con el AWS X-Ray demonio: utilícelo si no desea actualizar el código de la aplicación.
- Utilice el SDK de X-Ray con el CloudWatch agente: (recomendado) El CloudWatch agente es compatible con el SDK de X-Ray.

Para usar el SDK de X-Ray, haga lo siguiente:

- Instrumente su aplicación con el SDK de X-Ray.
- Configure un recopilador para que le indique dónde enviar los datos que recopila. Puede utilizar el CloudWatch agente o el daemon de X-Ray para recopilar la información de rastreo.

Una vez que el recopilador o el agente reciben los datos, los envían a un AWS servidor que usted especifique en la configuración del agente. El SDK de X-Ray solo puede enviar datos a un backend de AWS , tal como se muestra en el siguiente diagrama:



Si está utilizando Java, puede emplear el SDK de X-Ray para instrumentar automáticamente su aplicación. Para empezar a usar el SDK de X-Ray, consulte las bibliotecas asociadas a los siguientes lenguajes de programación:

- Go
- Java

- Node.js
- Python
- .NET
- Ruby

Uso de una consola

Utilice una consola si quiere emplear una interfaz gráfica de usuario (GUI) que requiera un mínimo de programación. Los usuarios que acaban de comenzar en X-Ray pueden ponerse manos a la obra rápidamente mediante las visualizaciones prediseñadas con las que llevar a cabo tareas básicas. Puede hacer lo siguiente directamente desde la consola:

- Habilite X-Ray.
- Consultar los resúmenes de alto nivel del rendimiento de su aplicación.
- Comprobar el estado de sus aplicaciones.
- Identificar errores de alto nivel.
- Consultar los resúmenes de rastreo básicos.

Para interactuar con X-Ray, puede utilizar tanto la consola de Amazon CloudWatch en https://console.aws.amazon.com/cloudwatch/ como la consola de X-Ray en https://console.aws.amazon.com/xray/home.

Uso de la consola de Amazon CloudWatch

La consola de CloudWatch incluye una nueva funcionalidad de X-Ray que se ha rediseñado a partir de la consola X-Ray para facilitar su uso. Si usa la consola de CloudWatch, puede ver los registros y las métricas de CloudWatch junto con los datos de rastros de X-Ray. Utilice la consola de CloudWatch para ver y analizar datos, incluidos los siguientes:

- Rastros de X-Ray: vea, analice y filtre los rastros asociados a su aplicación mientras se atiende una solicitud. Utilice estos rastros para encontrar latencias altas, depurar errores y optimizar el flujo de trabajo de su aplicación. Consulte mapas de rastros y de servicios para ver representaciones visuales del flujo de trabajo de su aplicación.
- Registros: consulte, analice y filtre los registros que produce la aplicación. Utilice los registros para solucionar errores y configurar la supervisión en función de valores de registro específicos.

- Métricas: calcule y supervise el rendimiento de su aplicación mediante las métricas que emiten sus recursos o cree sus propias métricas. Consulte estas métricas en gráficos o diagramas.
- Supervisión de redes e infraestructuras: supervise las redes principales para detectar interrupciones y comprobar el estado y el rendimiento de su infraestructura, incluidas las aplicaciones en contenedores, otros servicios de AWS y los clientes.
- Todas las funciones de la consola de X-Ray que se enumeran en la siguiente sección Uso de la consola de X-Ray.

Para obtener más información sobre la consola de CloudWatch, consulte Introducción a Amazon CloudWatch.

Inicie sesión en la consola de Amazon CloudWatch desde https://console.aws.amazon.com/cloudwatch/.

Uso de la consola de X-Ray

La consola de X-Ray ofrece un rastreo distribuido para las solicitudes de aplicaciones. Utilice la consola de X-Ray si quiere trabajar con una consola de la forma más sencilla o si no quiere actualizar el código de la aplicación. AWS ha dejado de desarrollar la consola de X-Ray. La consola de X-Ray contiene las siguientes características para aplicaciones instrumentadas:

- Información: detecta automáticamente las anomalías en el rendimiento de su aplicación y encuentra las causas subyacentes. Esta información se incluye en la consola de CloudWatch, en la sección Información. Para obtener más información, consulte Uso de Información de X-Ray en la Uso de la consola de X-Ray.
- Mapa de servicios: vea una estructura gráfica de su aplicación y sus conexiones con los clientes, los recursos, los servicios y las dependencias.
- Rastros: consulte un resumen de los rastros que genera su aplicación cuando atiende una solicitud. Utilice los datos de rastro para comprender el rendimiento de su aplicación en comparación con las métricas básicas, como la respuesta HTTP y el tiempo de respuesta.
- Análisis: interprete, explore y analice los datos de rastro mediante gráficos para distribuir el tiempo de respuesta.
- Configuración: cree rastros personalizados para cambiar las configuraciones predeterminadas de lo siguiente:

- Muestreo: cree una regla que defina la frecuencia con la que debe tomar una muestra de su solicitud para recibir la información de rastro. Para obtener más información, consulte Configuración de reglas de muestreo en la Uso de la consola de X-Ray.
- Cifrado: cifra los datos en reposo con una clave que se puede auditar o deshabilitar con AWS Key Management Service.
- Grupos: utilice una expresión de filtro para definir un grupo de rastros con una característica común, como el nombre de una URL o el tiempo de respuesta. Para obtener más información, consulte Configuración de grupos.

Inicie sesión en la consola de X-Ray en https://console.aws.amazon.com/xray/home.

Exploración de la consola de X-Ray

Utilice la consola de X-Ray para ver un mapa de los servicios y rastros asociadas a las solicitudes que atienden sus aplicaciones, y para configurar grupos y reglas de muestreo que afecten a la forma en que se envían los rastros a X-Ray.

> **Note**
>
> El mapa de servicio X-Ray y el mapa ServiceLens de CloudWatch se han combinado en el mapa de rastros de X-Ray de la consola de Amazon CloudWatch. Abra la consola de CloudWatch y seleccione Mapa de rastros en Rastros de X-Ray, en el panel de navegación izquierdo.
>
> Ahora, CloudWatch se integra con Application Signals, que puede detectar y supervisar los servicios de aplicaciones, los clientes, los canarios de Synthetics y las dependencias de los servicios. Use Application Signals para ver una lista o un mapa visual de sus servicios, ver las métricas del estado en función de los objetivos de nivel de servicio (SLO) y profundizar para ver los seguimientos de X-Ray correlacionados para una solución de problemas más detallada.

La página principal de la consola principal de X-Ray es el mapa de rastros, que es una representación visual del gráfico de servicios de JSON que X-Ray genera a partir de los datos de rastro que sus aplicaciones generan. El mapa se compone de nodos de servicios para cada aplicación en su cuenta que atiende solicitudes, nodos cliente principales que representan los orígenes de las solicitudes, y nodos de servicios posteriores que representan los servicios web y los

recursos utilizados por una aplicación mientras procesa una solicitud. Hay páginas adicionales para ver rastros y sus detalles, y para configurar grupos y reglas de muestreo.

Vea la experiencia de la consola de X-Ray y compárela con la consola de CloudWatch en las siguientes secciones.

Descubra las consolas de X-Ray y CloudWatch

- Uso del mapa de rastros de X-Ray
- Visualización de rastros y detalles de rastros
- Uso de expresiones de filtro
- Rastreo entre cuentas
- Rastreo de aplicaciones basadas en eventos
- Uso de histogramas de latencia
- Uso de información en X-Ray
- Interacción con la consola de Analytics
- Configuración de grupos
- Configuración de reglas de muestreo de
- Configuración de muestreo flexible
- Enlace profundo de la consola

Uso del mapa de rastros de X-Ray

Utilice el mapa de rastros en la consola de X-Ray para identificar los servicios donde ocurran errores, donde haya conexiones con alta latencia o rastros de solicitudes que dieron error.

> **Note**
>
> Ahora, CloudWatch se integra con Application Signals, que puede detectar y supervisar los servicios de aplicaciones, los clientes, los canarios de Synthetics y las dependencias de los servicios. Use Application Signals para ver una lista o un mapa visual de sus servicios, ver las métricas del estado en función de los objetivos de nivel de servicio (SLO) y profundizar para ver los seguimientos de X-Ray correlacionados para una solución de problemas más detallada.
> El mapa de servicio X-Ray y el mapa ServiceLens de CloudWatch se han combinado en el mapa de rastros de X-Ray de la consola de Amazon CloudWatch. Abra la consola de

CloudWatch y seleccione Mapa de rastros en Rastros de X-Ray, en el panel de navegación izquierdo.

Consulta del mapa de rastros

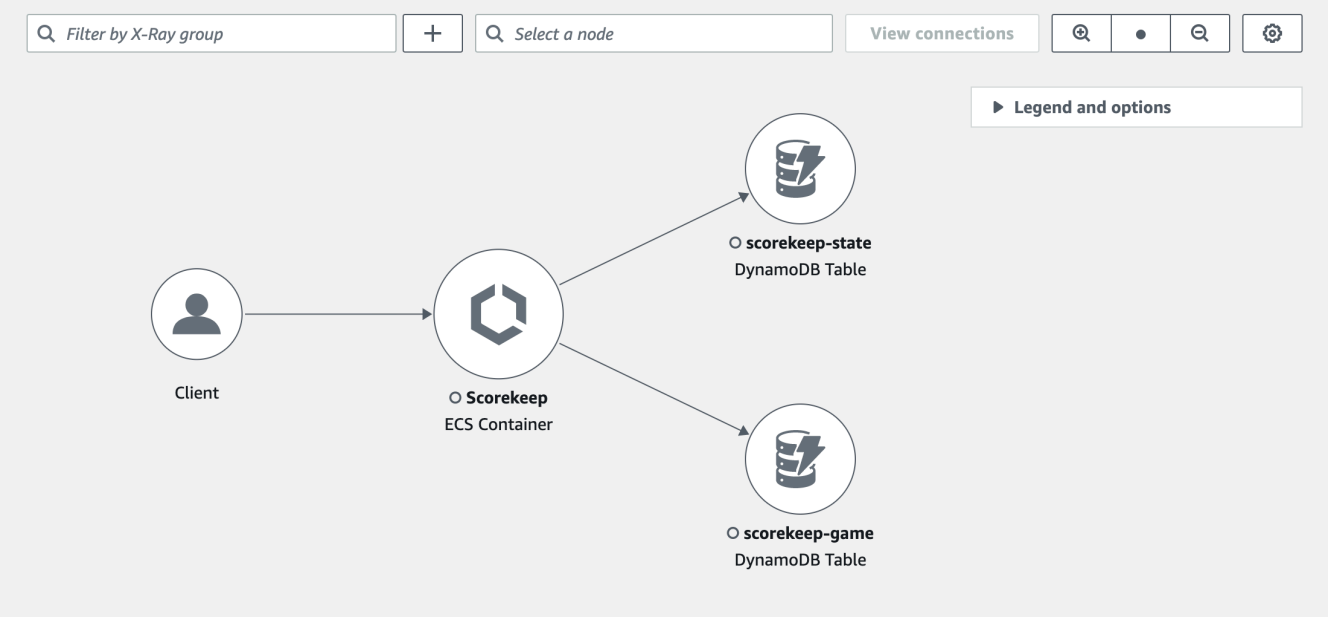
El mapa de rastros es una representación visual de los datos de rastro que sus aplicaciones generan. El mapa muestra nodos de servicios que atienden solicitudes, nodos cliente principales que representan los orígenes de las solicitudes, y nodos de servicios posteriores que representan los servicios web y los recursos utilizados porque utiliza una aplicación mientras procesa una solicitud.

El mapa de rastros muestra una vista conectada de los rastros de las aplicaciones basadas en eventos que utilizan Amazon SQS y Lambda. Para obtener más información, consulte Rastreo de aplicaciones basadas en eventos. El mapa de rastros también admite el rastreo entre cuentas, ya que muestra los nodos de varias cuentas en un solo mapa.

CloudWatch console

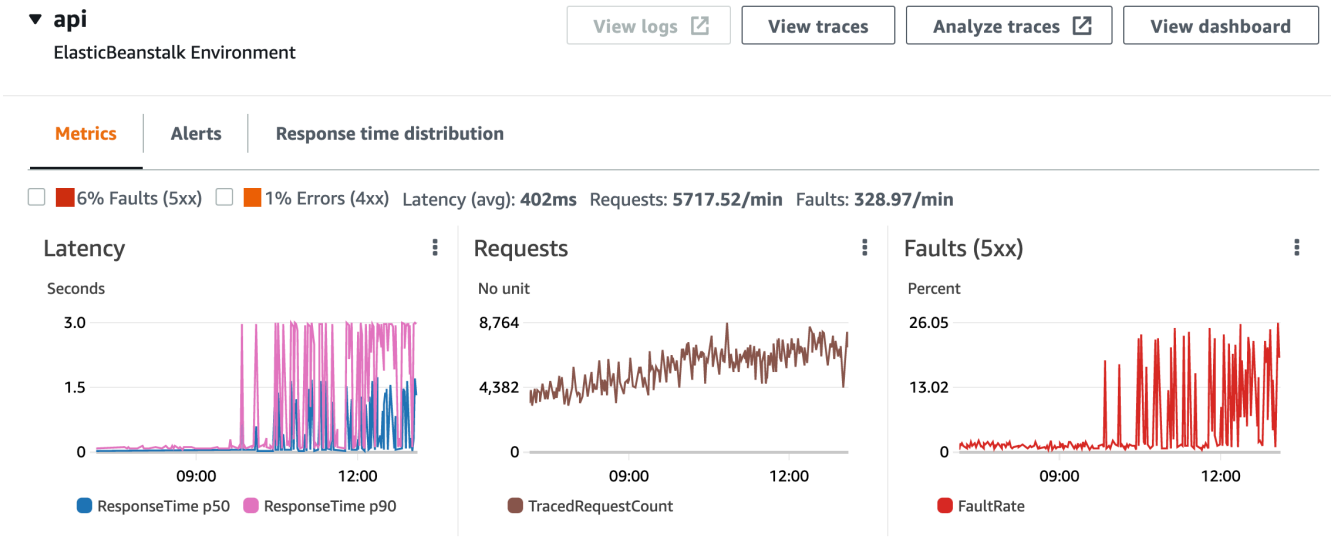
Visualización del mapa de rastros en la consola CloudWatch

1. Abra la consola de CloudWatch. Elija mapa de rastros en la sección Rastros de X-Ray en el panel de navegación izquierdo.



2. Seleccione un nodo de servicio para ver sus solicitudes o un límite entre dos nodos para ver las solicitudes que pasaron por esa conexión.

3. Debajo del mapa de rastros se muestra información adicional, como pestañas para las métricas, alertas y la distribución del tiempo de respuesta. En la pestaña Métricas, seleccione un rango dentro de cada gráfico para desglosarlo y ver más detalles, o elija las opciones Fallos o Errores para filtrar los rastros. En la pestaña Distribución del tiempo de respuesta, seleccione un rango dentro del gráfico para filtrar los rastros por tiempo de respuesta.



4. Para ver los rastros, elija Ver rastros o, si ha aplicado un filtro, elija Ver rastros filtrados.
5. Seleccione Ver registros para ver los registros de CloudWatch asociados al nodo seleccionado. No todos los nodos del mapa de rastros permiten ver los registros. Consulte la solución de problemas de los registros de CloudWatch para obtener más información.

El mapa de rastros indica los problemas en cada nodo delineándolo con colores:

- El rojo se usa para fallos de servidor (errores de la serie 500)
- El amarillo se usa para los errores del cliente (errores de la serie 400)
- El morado indica los errores de limitación de solicitudes (429: demasiadas solicitudes)

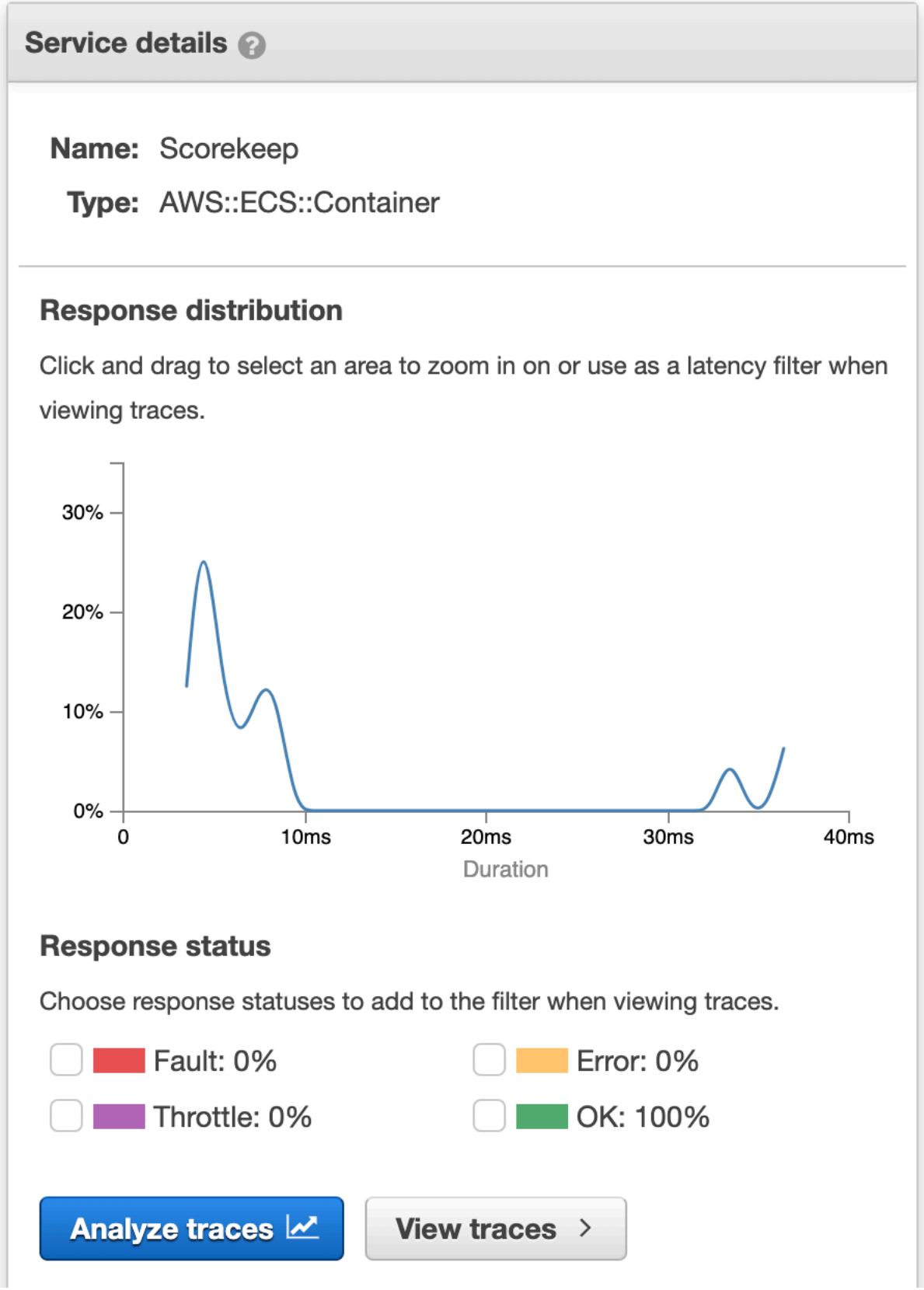
Si el mapa de rastros es demasiado grande, utilice los controles que aparecen en pantalla o el ratón para ampliar y reducir el mapa o para moverlo por la pantalla.

X-Ray console

Visualización del mapa de servicio

1. Abra la consola de X-Ray. El mapa de servicio se muestra de forma predeterminada. También puede elegir mapa de servicio en el panel de navegación izquierdo.

2. Seleccione un nodo de servicio para ver sus solicitudes o un límite entre dos nodos para ver las solicitudes que pasaron por esa conexión.
3. Utilice el histograma que representa la distribución del tiempo de respuesta para filtrar los rastros por duración y seleccione los códigos de estado cuyos rastros desee ver. Después elija Ver rastros para abrir la lista de rastros con la expresión de filtro aplicada.


Service details
Name: Scorekeep
Type: AWS::ECS::Container
Response distribution
Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.
30%
20%
10%
0%
0
10ms
20ms
30ms
40ms
Duration
Response status
Choose response statuses to add to the filter when viewing traces.
Fault: 0%
Error: 0%
Throttle: 0%
OK: 100%
Analyze traces
View traces

El mapa de servicio indica el estado de cada nodo mediante el uso de colores en función de la proporción de llamadas correctas y errores o fallos:

- El verde se utiliza para las llamadas realizadas con éxito
- El rojo se usa para fallos de servidor (errores de la serie 500)
- El amarillo se usa para los errores del cliente (errores de la serie 400)
- El morado indica los errores de limitación de solicitudes (429: demasiadas solicitudes)

Si el mapa de servicio es demasiado grande, utilice los controles que aparecen en pantalla o el ratón para ampliar y reducir el mapa o para moverlo por la pantalla.

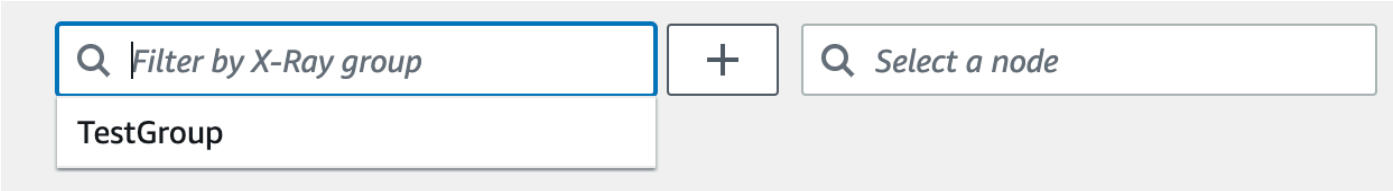
> **Note**
>
> El mapa de rastros de X-Ray puede mostrar hasta 10 000 nodos. En raras ocasiones, en las que el número total de nodos de servicio supere este límite, es posible que reciba un mensaje de error reciba y no pueda visualizar un mapa de rastros completo en la consola.

Filtrado del mapa de rastros por grupo

Mediante una expresión de filtro, puede definir los criterios por los que incluir rastros en un grupo. Siga los pasos que se indican a continuación para mostrar ese grupo específico en el mapa de rastros.

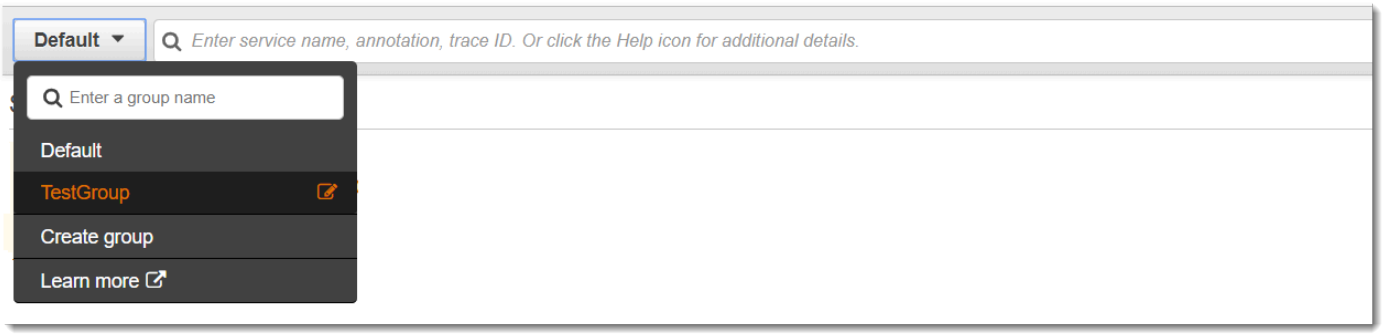
CloudWatch console

Elija un nombre de grupo en el filtro de grupo situado en la parte superior izquierda del mapa de rastros.



X-Ray console

Elija un nombre de grupo en el menú desplegable situado a la izquierda de la barra de búsqueda.



Entonces se filtrará el mapa de servicio para mostrar los rastros que coincidan con la expresión de filtro del grupo seleccionado.

Leyenda y opciones del mapa de rastros

El mapa de rastros incluye una leyenda y varias opciones para personalizar la visualización del mapa.

CloudWatch console

Seleccione el menú desplegable Leyenda y opciones en la parte superior derecha del mapa. Elija lo que se muestra dentro de los nodos, por ejemplo:

- Métricas muestra el tiempo medio de respuesta y el número de rastros enviados por minuto durante el intervalo de tiempo elegido.
- Nodos muestra el icono de servicio dentro de cada nodo.

Seleccione otros ajustes del mapa en el panel Preferencias, al que se puede acceder mediante el icono con forma de engranaje situado en la parte superior derecha del mapa. Estos ajustes incluyen seleccionar qué métrica se utiliza para determinar el tamaño de cada nodo y qué valores controlados deben mostrarse en el mapa.

X-Ray console

Para ver la leyenda del mapa de servicio, seleccione el enlace Leyenda del mapa en la parte superior derecha del mapa. Las siguientes opciones se pueden elegir en la parte inferior derecha del mapa de rastros:

- Iconos de servicio cambia lo que se muestra en cada nodo y muestra el icono del servicio o el tiempo medio de respuesta y el número de rastros enviados por minuto durante el intervalo de tiempo elegido.
- Tamaño de los nodos: ninguno establece el mismo tamaño para todos los nodos.
- Tamaño de los nodos: estado dimensiona los nodos en función del número de solicitudes afectadas por errores, fallos o solicitudes limitadas.
- Tamaño de los nodos: tráfico dimensiona los nodos según el número total de solicitudes.

Visualización de rastros y detalles de rastros

Utilice la página Rastros de la consola de X-Ray para encontrar rastros por URL, código de respuesta u otros datos a partir del resumen de rastros. Tras seleccionar un rastro de la lista de rastros, la página de Detalles de rastreo muestra un mapa de los nodos de servicio asociados con el rastro seleccionado, junto con una escala de tiempo de los segmentos del rastro.

Consulta de registros de seguimiento

CloudWatch console

Para ver rastros en la consola de CloudWatch

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación izquierdo, seleccione Rastros de X-Ray y, a continuación, Rastros. Puede filtrar por grupo o introducir una expresión de filtro. De esta manera, se filtran los rastros que aparecen en la sección Rastros, en la parte inferior de la página.

   También puede utilizar el mapa de servicio para ir a un nodo de servicio específico y, a continuación, ver los rastros. Esto abre la página de rastros con una consulta ya aplicada.
3. Acote su consulta en la sección Limitadores de consultas. Para filtrar los rastros por un atributo común, elija una opción en la flecha hacia abajo situada junto a Ajustar consulta por. Las opciones incluyen:

   - Nodo: filtra los rastros por nodo de servicio.
   - ARN del recurso: filtra los rastros por un recurso asociado a un rastro. Algunos ejemplos de estos recursos incluyen una instancia de Amazon Elastic Compute Cloud (Amazon EC2), una función de AWS Lambda o una tabla de Amazon DynamoDB.

- Usuario: filtre los rastros con un ID de usuario.
- Mensaje de causa raíz del error: filtra los rastros por la causa raíz del error.
- URL: filtra los rastros por la ruta URL utilizada por la aplicación.
- Código de estado HTTP: filtra los rastros por el código de estado HTTP devuelto por la aplicación. Puede especificar un código de respuesta personalizado o seleccionar una de las siguientes opciones:
  - `200`: la solicitud se ha realizado correctamente.
  - `401`: la solicitud carecía de credenciales de autenticación válidas.
  - `403`: la solicitud carecía de permisos válidos.
  - `404`: el servidor no pudo encontrar el recurso solicitado.
  - `500`: el servidor detectó una condición inesperada y generó un error interno.

Elija uno o más valores y seleccione Agregar a consulta para añadirlos a la expresión de filtro de la parte superior de la página.

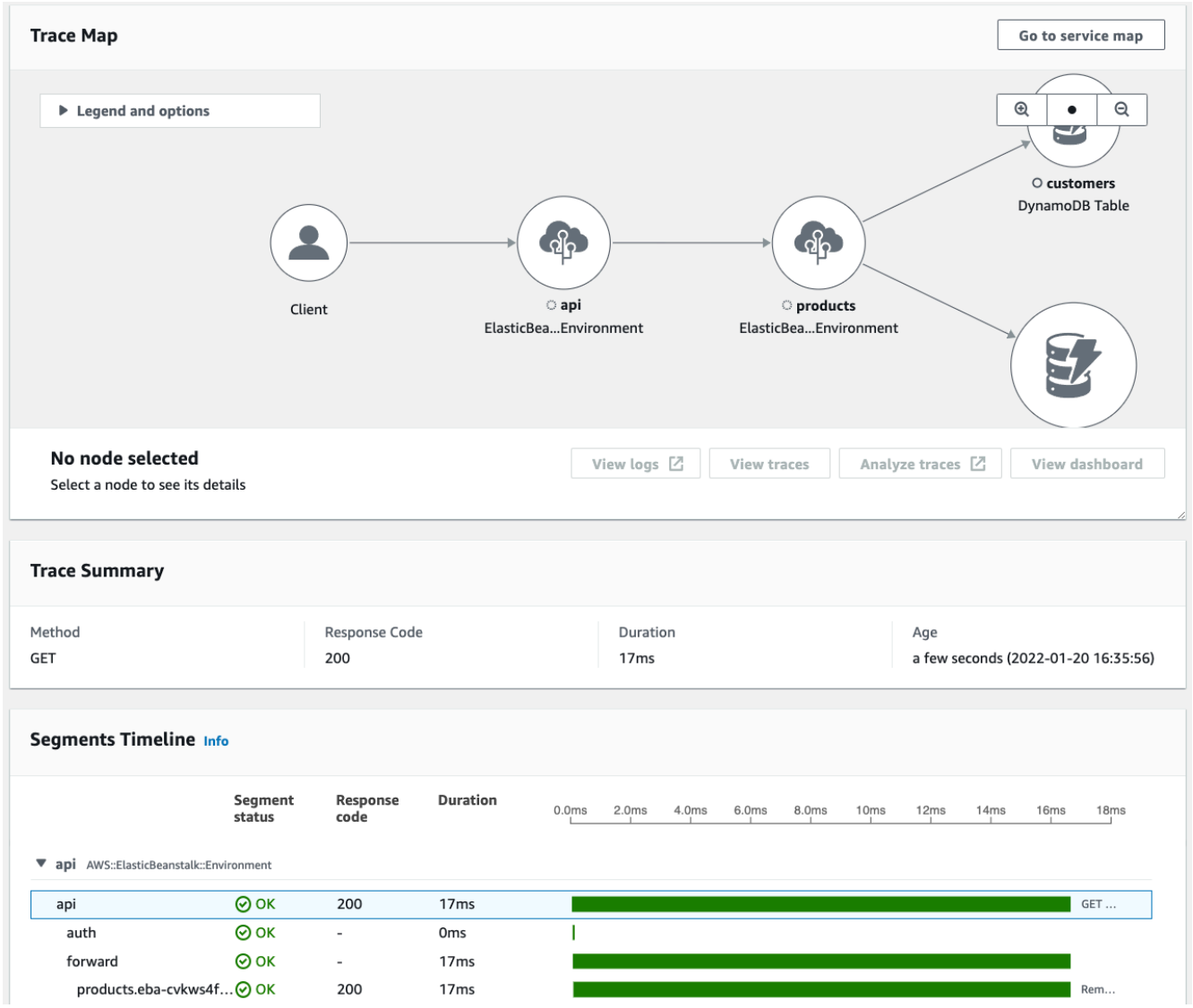
4. Para encontrar un único rastro, escriba un ID de rastro directamente en el campo de consulta. Puede utilizar el formato de X-Ray o el formato del World Wide Web Consortium (W3C). Por ejemplo, un rastro que se crea con AWS Distro para OpenTelemetry tendrá el formato W3C.

   > **Note**
   >
   > Al consultar rastros creados con un ID de rastro en formato W3C, la consola mostrará el rastro coincidente en formato X-Ray, por ejemplo. Por ejemplo, si realiza una consulta para `4efaaf4d1e8720b39541901950019ee5` en formato W3C, la consola muestra el equivalente de X-Ray: `1-4efaaf4d-1e8720b39541901950019ee5`.

5. Seleccione Ejecutar consulta en cualquier momento para visualizar una lista de rastros coincidentes en la sección Rastros, en la parte inferior de la página.
6. Elija un ID de rastro de la lista para mostrar la página Detalles de la traza de un solo rastro.

   La siguiente imagen muestra un mapa de rastros que contiene los nodos de servicio asociados al rastro y la periferia entre los nodos que representan la ruta seguida por los segmentos que componen el rastro. Tras Mapa de rastros, aparece Resumen de los rastros. El resumen contiene información sobre una operación `GET` de ejemplo, su código

de respuesta, la duración del rastreo y la edad de la solicitud. Tras Resumen de los rastros aparece Línea temporal de los segmentos, que muestra la duración de los segmentos y subsegmentos del rastro.



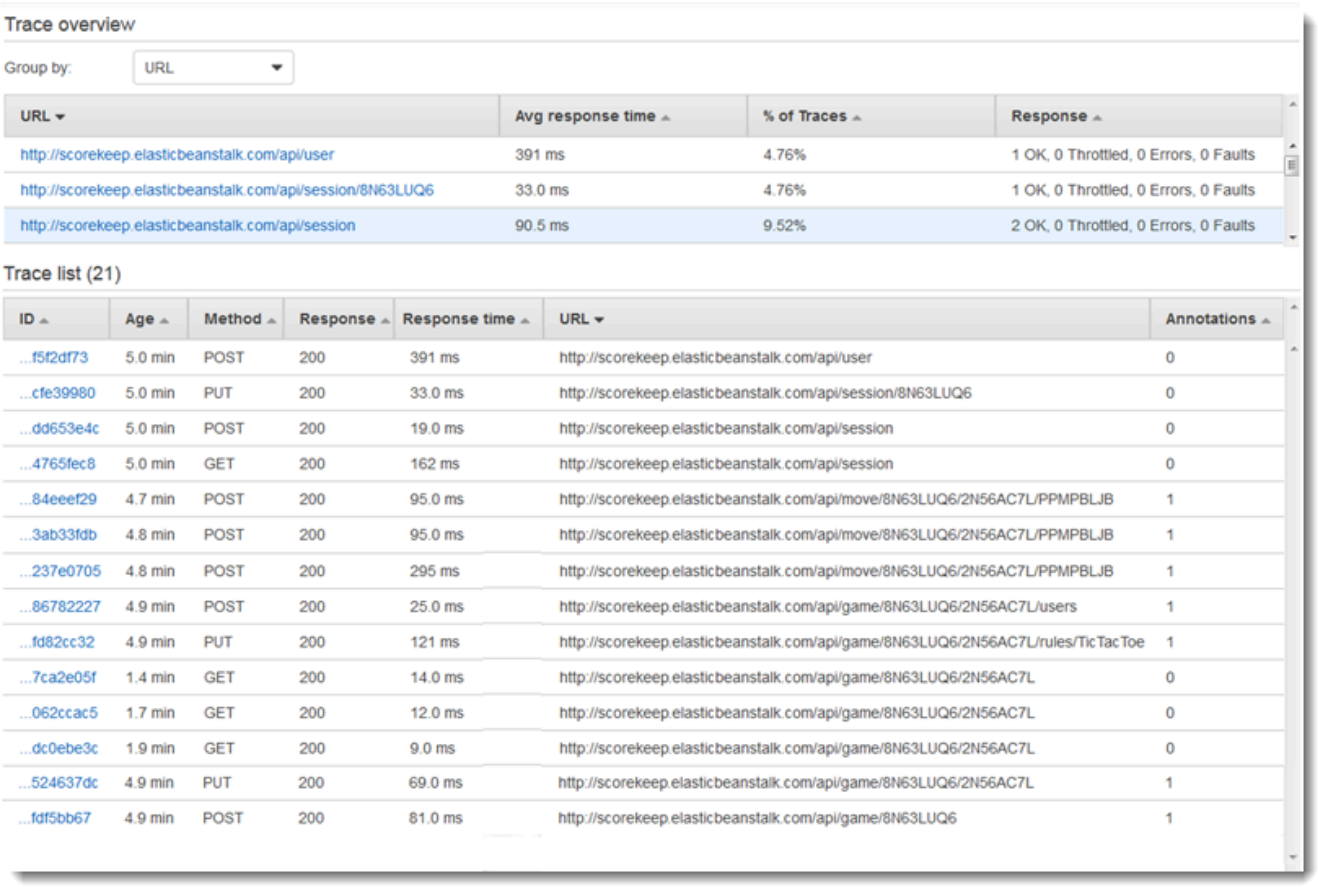
Si tiene una aplicación basada en eventos que utiliza Amazon SQS y Lambda, puede ver una vista conectada de los rastros de cada solicitud en el mapa de rastros. En el mapa, los rastros de los productores de los mensajes están vinculados a los de los consumidores de AWS Lambda y se muestran como un borde discontinuo. Para obtener más información acerca de las aplicaciones basadas en eventos, consulte Rastreo de aplicaciones basadas en eventos.

Las páginas Rastros y Detalles de rastros también admiten el rastreo entre cuentas, ya que muestran los rastros de varias cuentas en la lista de rastros y en un solo mapa.

X-Ray console

Para ver los rastros en la consola de X-Ray

1. Abra la página Rastros de la consola de X-Ray. El panel Información general del rastro muestra una lista de rastros agrupados por características comunes, como las causas raíz de los errores, el ARN de recurso o el InstanceID.
2. Para seleccionar una característica común y ver un conjunto agrupado de rastros, expanda la flecha hacia abajo situada junto a Agrupar por. La siguiente ilustración muestra la información general de los rastros agrupados por URL en AWS X-Ray ejemplo de aplicación y una lista de los rastros asociados.



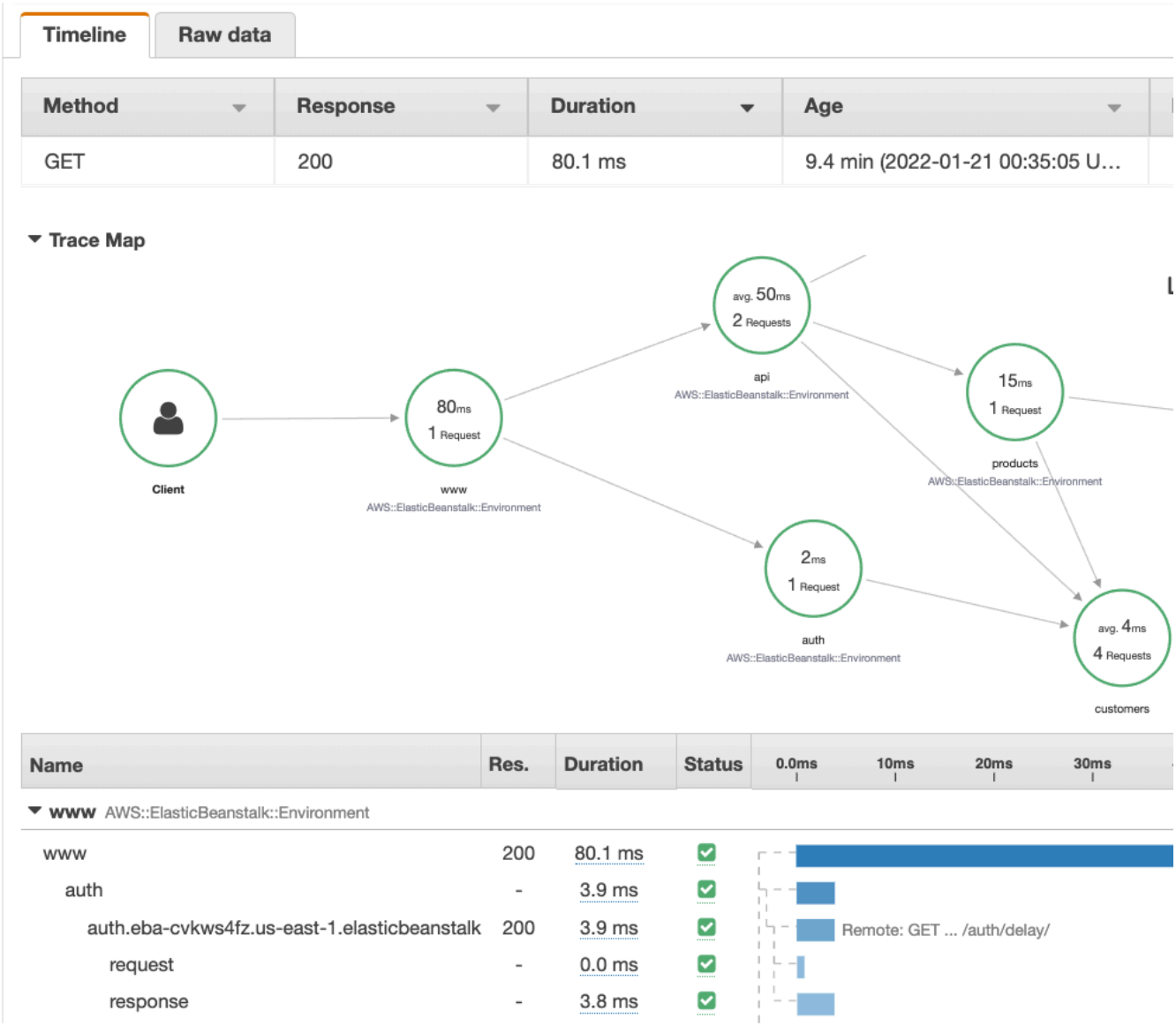
3. Elija el ID de un rastro para verlo en la lista de rastros. También puede elegir Mapa de servicios en el panel de navegación para ver los rastros de un nodo de servicio en particular. A continuación, puede ver los rastros asociados a ese nodo.

   La pestaña Plazo muestra el flujo de solicitudes del rastro e incluye lo siguiente:

   - Un mapa de la ruta de cada segmento del rastro.

- Cuánto tiempo tarda el segmento en llegar a un nodo del mapa de rastros.
- Cuántas solicitudes se realizaron al nodo del mapa de rastros.

En la siguiente ilustración se muestra un ejemplo de mapa de rastros asociado a una solicitud `GET` efectuada a una aplicación de ejemplo. Las flechas muestran la ruta que siguió cada segmento para completar la solicitud. Los nodos de servicio muestran el número de solicitudes realizadas durante la solicitud `GET`.



Para obtener más información sobre la pestaña Plazo, consulte la siguiente sección Exploración de la escala de tiempo del rastro.

La pestaña de Datos sin formato muestra información sobre el rastro y los segmentos y subsegmentos que lo componen, en formato `JSON`. Esta información puede incluir lo siguiente:

- Marcas temporales
- ID únicos
- Recursos asociados al segmento o subsegmento
- La fuente o el origen del segmento o subsegmento
- Información adicional sobre la solicitud a su aplicación, como la respuesta de una solicitud HTTP

Exploración de la escala de tiempo del rastro

La sección Plazo muestra una jerarquía de segmentos y subsegmentos junto a una barra horizontal que corresponde al tiempo que tardaron en completar sus tareas. La primera entrada de la lista es el segmento, que representa todos los datos registrados por el servicio para una misma solicitud. Los subsegmentos se representan tras una sangría y se enumeran a continuación del segmento. Las columnas contienen información sobre cada segmento.

CloudWatch console

En la consola de CloudWatch, Línea temporal de los segmentos proporciona la siguiente información:

- La primera columna muestra los segmentos y subsegmentos del rastro seleccionado.
- La columna Estado del segmento muestra el resultado del estado de cada segmento y subsegmento.
- La columna Código de respuesta muestra un código de estado de respuesta HTTP a una solicitud del navegador realizada por el segmento o subsegmento, cuando está disponible.
- La columna Duración muestra cuánto tiempo duró la ejecución del segmento o subsegmento.
- La columna Alojado en muestra el espacio de nombre o el entorno en el que se ejecuta el segmento o subsegmento, si corresponde. Para obtener más información, consulte Dimensions collected and dimension combinations.

- La última columna muestra barras horizontales que corresponden a la duración de la ejecución del segmento o subsegmento, en relación con los demás segmentos o subsegmentos de la línea de tiempo.

Para agrupar la lista de segmentos y subsegmentos por nodo de servicio, active Agrupar por nodos.

X-Ray console

En la página de detalles del rastro, seleccione la pestaña Línea temporal para ver la línea de tiempo de cada segmento y subsegmento que forma parte de un rastro.

En la consola de X-Ray, Plazo proporciona la siguiente información:

- La columna Nombre muestra los nombres de los segmentos y subsegmentos del rastro.
- La columna Res. muestra un código de estado de respuesta HTTP a una solicitud del navegador realizada por el segmento o subsegmento, cuando está disponible.
- La columna Duración muestra cuánto tiempo duró la ejecución del segmento o subsegmento.
- La columna Estado muestra el resultado del estado del segmento o subsegmento.
- La última columna muestra barras horizontales que corresponden a la duración de la ejecución del segmento o subsegmento, en relación con los demás segmentos o subsegmentos de la línea de tiempo.

Para ver los datos de rastro sin procesar que la consola utiliza para generar la línea de tiempo, seleccione la pestaña Datos sin formato. Los datos sin procesar muestran información sobre el rastro y los segmentos y subsegmentos que lo componen en formato `JSON`. Esta información puede incluir lo siguiente:

- Marcas temporales
- ID únicos
- Recursos asociados al segmento o subsegmento
- La fuente o el origen del segmento o subsegmento
- Información adicional sobre la solicitud a su aplicación, como la respuesta de una solicitud HTTP.

Cuando se utiliza un cliente del SDK de AWS, HTTP o SQL para realizar llamadas a recursos externos. el SDK de X-Ray registra los subsegmentos automáticamente. También puede emplear el SDK de X-Ray para registrar los subsegmentos personalizados de cualquier bloque de código o de funciones. Los subsegmentos adicionales que se registran mientras hay subsegmentos abiertos se convierten en elementos secundarios del subsegmento personalizado.

Consulta de detalles de segmentos

En la sección Plazo del rastro, elija el nombre de un segmento para ver sus detalles.

El panel Detalles del segmento muestra las pestañas Descripción general, Recursos, Anotaciones, Metadatos, Excepciones y SQL. Se aplica lo siguiente:

- En Overview (Información general) se muestra información acerca de la solicitud y la respuesta. La información incluye el nombre, la hora de inicio, la hora de finalización, la duración, la URL de la solicitud, la operación de la solicitud, el código de respuesta de la solicitud y cualquier error o fallo.
- La pestaña Recursos de un segmento muestra información del SDK de X-Ray y sobre los recursos de AWS que se utilizan para ejecutar la aplicación. Utilice los complementos Amazon EC2, AWS Elastic Beanstalk o Amazon ECS para que el SDK de X-Ray registre información sobre los recursos específicos de los servicios. Para obtener más información sobre complementos, consulte Complementos de servicio en la Configuración del SDK de X-Ray para Java.
- En las pestañas restantes (Anotaciones, Metadatos y Excepciones), se muestran las anotaciones, los metadatos y las excepciones que se registran en el segmento. Las excepciones se capturan automáticamente cuando las genera una solicitud instrumentada. Las anotaciones y los metadatos contienen información adicional que se registra utilizando las operaciones proporcionadas por el SDK de X-Ray. Para añadir anotaciones y metadatos a sus segmentos, use el SDK de X-Ray. Para obtener más información, consulte el enlace específico del idioma que aparece en la sección Cómo instrumentar su aplicación con los SDK de AWS X-Ray en Instrumentación de su solicitud para AWS X-Ray.

Consulta de detalles de subsegmentos

En la escala de tiempo del rastro, elija el nombre de un segmento para ver sus detalles:

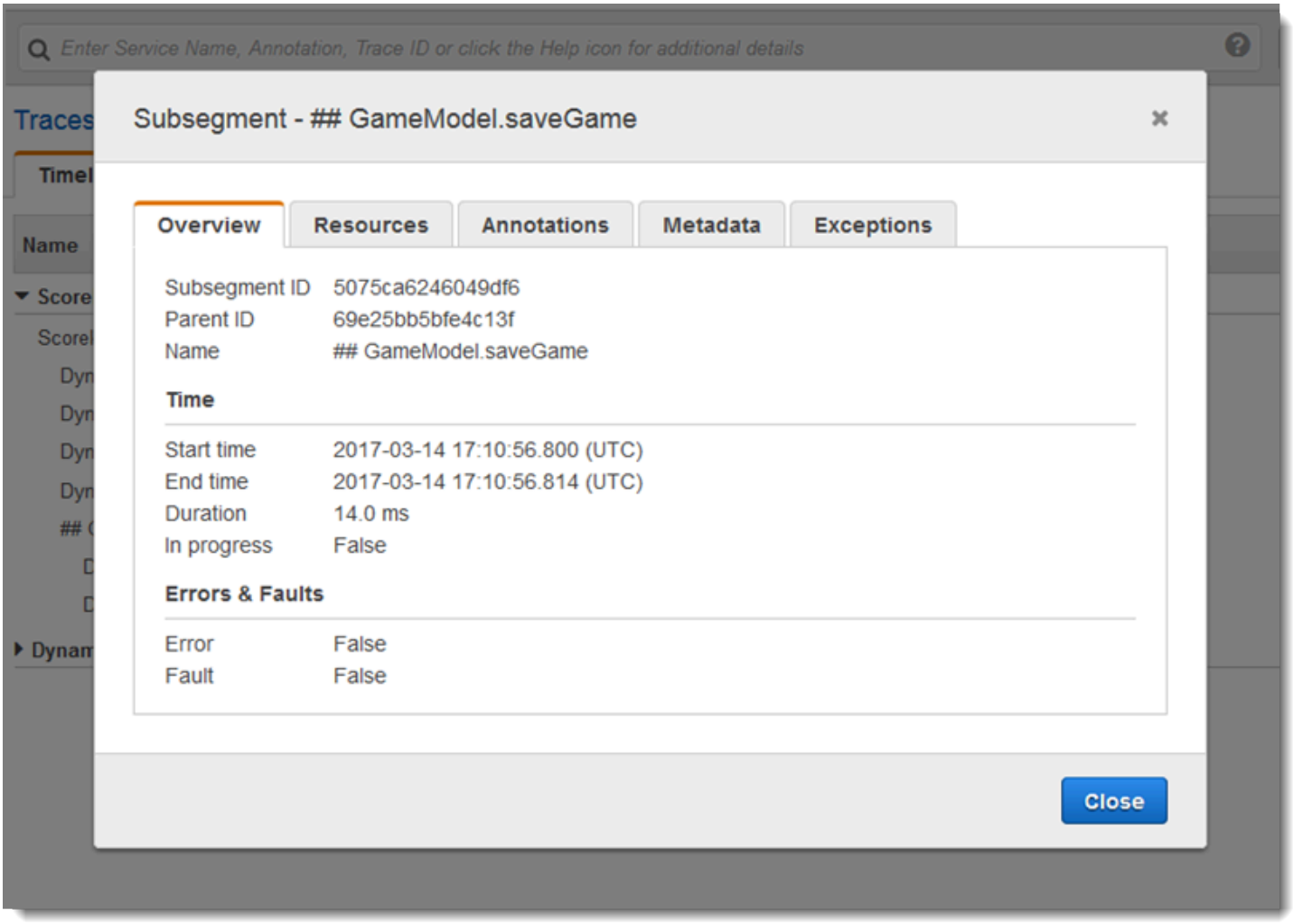
- En la pestaña Información general se muestra información acerca de la solicitud y su respuesta. Incluye el nombre, la hora de inicio, la hora de finalización, la duración, la solicitud URL, la operación de solicitud, el código de respuesta a la solicitud y cualquier error o fallo. Para los

subsegmentos generados con clientes instrumentados, la pestaña Overview (Información general) contiene información acerca de la solicitud y la respuesta desde el punto de vista de la aplicación.

- La pestaña Recursos de un subsegmento muestra detalles sobre los recursos de AWS que se emplearon para ejecutar el subsegmento. Por ejemplo, la pestaña de recursos puede incluir un ARN de función de AWS Lambda, información sobre una tabla de DynamoDB, cualquier operación a la que se llame o un ID de solicitud.
- En las pestañas restantes (Anotaciones, Metadatos y Excepciones), se muestran las anotaciones, los metadatos y las excepciones registrados en el subsegmento. Las excepciones se capturan automáticamente cuando las genera una solicitud instrumentada. Las anotaciones y los metadatos contienen información adicional que se registra utilizando las operaciones proporcionadas por el SDK de X-Ray. Utilice el SDK de X-Ray para agregar anotaciones o metadatos a sus segmentos. Para obtener más información, consulte el enlace específico del idioma que aparece en la sección Cómo instrumentar su aplicación con los SDK de AWS X-Ray en Instrumentación de su solicitud para AWS X-Ray.

En los subsegmentos personalizados, la pestaña Información general muestra el nombre del subsegmento, que se puede establecer de modo que especifique el área de código o la función que registra. Para obtener más información, consulte el enlace específico del idioma que aparece en la sección Cómo instrumentar su aplicación con los SDK de AWS X-Ray en Generación de subsegmentos personalizados con el SDK de X-Ray para Java.

La siguiente imagen muestra la pestaña Información general de un subsegmento personalizado. El resumen contiene el ID del subsegmento, el ID principal, el nombre, las horas de inicio y finalización, la duración, el estado y los errores o fallos.



La pestaña Metadatos de un subsegmento personalizado contiene información en formato JSON sobre los recursos utilizados por ese subsegmento.

Uso de expresiones de filtro

Utilice expresiones de filtro para ver un mapa de rastros o los rastros de una solicitud específica, un servicio específico o una conexión entre dos servicios (una periferia) o solicitudes que respondan a una condición. X-Ray proporciona un lenguaje de expresiones de filtro para filtrar solicitudes, servicios y periferias teniendo en cuenta los datos presentes en los encabezados de la solicitud, el estado de la respuesta y los campos indexados en los segmentos originales.

Al elegir un periodo de tiempo de registros de seguimiento para ver en la consola de X-Ray, puede obtener más resultados que los que la consola puede mostrar. En la esquina superior derecha, la consola muestra el número de registros de seguimiento que analiza y si hay más registros de

seguimiento disponibles. Puede utilizar una expresión de filtro para reducir el número de resultados a tan solo los rastros que desea encontrar.

Temas

Detalles de expresión de filtro

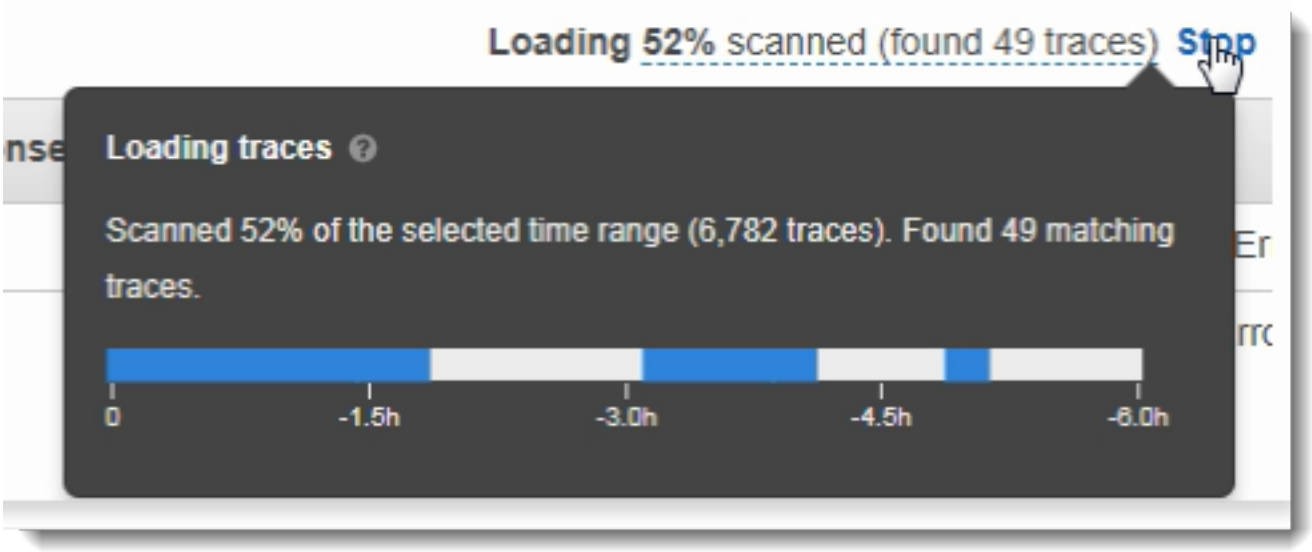
Cuando elige un nodo en el mapa de rastros, la consola construye una expresión de filtro en función del nombre de servicio del nodo y los tipos de errores presentes en función de su selección. Para encontrar registros de seguimiento que muestren problemas de rendimiento o relacionados con solicitudes específicas, puede ajustar la expresión proporcionada por la consola, o bien crear la suya propia. Si añade anotaciones con el SDK de X-Ray, también puede filtrar en función de la presencia de una clave de anotación o el valor de una clave.

> **Note**
>
> Si elige un intervalo de tiempo relativo en el mapa de rastros y elige un nodo, la consola convierte el intervalo de tiempo a una hora absoluta de inicio y finalización. Para asegurarse de que los registros de seguimiento del nodo aparezcan en los resultados de búsqueda y evitar los tiempos de examen cuando el nodo no estaba activo, el intervalo de tiempo solo incluye las horas a las que el nodo envió registros de seguimiento. Si desea buscar con relación a la hora actual, puede volver a un intervalo de tiempo relativo en la página de registros de seguimiento y volver a analizar.

Si sigue habiendo más resultados disponibles de los que la consola puede mostrar, la consola muestra cuántos registros de seguimiento han coincidido y el número de registros de seguimiento

analizados. El porcentaje que se muestra es el porcentaje del marco temporal seleccionado que se analizó. Para asegurarse de ver todos los registros de seguimiento coincidentes representados en los resultados, filtre aún más la expresión de filtro o seleccione un marco temporal más corto.

Para obtener los resultados más recientes en primer lugar, la consola comienza a analizar al final del intervalo de tiempo y va hacia atrás. Si hay un gran número de registros de seguimiento, pero pocos resultados, la consola divide el intervalo de tiempo en porciones y las analiza en paralelo. La barra de progreso muestra las partes del intervalo de tiempo que se han analizado.



Uso de expresiones de filtro con grupos

Los grupos son una colección de rastros que se definen mediante una expresión de filtro. Puede utilizar grupos para generar gráficos de servicios adicionales y suministrar métricas de Amazon CloudWatch.

Los grupos se identifican por su nombre o un nombre de recurso de Amazon (ARN) y contienen una expresión de filtro. El servicio compara los registros de seguimiento de entrada con la expresión y los almacena en consecuencia.

Puede crear y modificar grupos mediante el menú desplegable situado a la izquierda de la barra de búsqueda de expresiones de filtro.

> **Note**
>
> Si el servicio encuentra un error al evaluar un grupo, ese grupo deja de estar incluido en el procesamiento de los registros de seguimiento de entrada y se registra una métrica de error.

Para obtener más información acerca de los grupos, consulte Configuración de grupos.

Sintaxis de expresiones de filtro

Las expresiones de filtro pueden incluir una palabra clave, un operador único o binario y un valor para la comparación.

```
keyword operator value
```

Cada tipo de palabra clave tiene sus propios operadores. Por ejemplo, `responsetime` es una palabra clave numérica que se puede comparar con operadores relacionados con números.

Example– solicitudes con tiempo de respuesta de más de 5 segundos

```
responsetime > 5
```

Puede combinar varias expresiones en una expresión compuesta utilizando los operadores `AND` u `OR`.

Example– solicitudes con duración total de 5 a 8 segundos

```
duration >= 5 AND duration <= 8
```

Solo se detectan problemas con las palabras clave y los operadores sencillos en el nivel de registro de seguimiento. Si se produce un error posterior, pero la aplicación lo gestiona y no se muestra al usuario, este problema no se detectará al buscar con la palabra `error`.

Para encontrar registros de seguimiento con problemas posteriores, use las palabras clave complejas `service()` y `edge()`. Estas palabras clave permiten aplicar un filtro de expresión a todos los nodos posteriores, a un único nodo posterior o a un límite entre dos nodos. Para obtener un mayor nivel de detalle, puede filtrar los servicios y los bordes con la función id().

Palabras clave booleanas

Los valores de palabra clave booleana son true o false. Utilícelas para encontrar registros de seguimiento que resultaron erróneos.

Palabras clave booleanas

- `ok`: el código de estado de la respuesta fue 2XX Success.

- `error`: el código de estado de la respuesta fue 4XX Client Error.
- `throttle`: el código de estado de la respuesta fue 429 Too Many Requests.
- `fault`: el código de estado de la respuesta fue 5XX Server Error.
- `partial`: la solicitud tiene segmentos incompletos.
- `inferred`: la solicitud tiene segmentos inferidos.
- `first`: el elemento es el primero de una lista enumerada.
- `last`: el elemento es el último de una lista enumerada.
- `remote`: la entidad de causa raíz es remota.
- `root`: la solicitud es el punto de entrada o el segmento raíz de un registro de seguimiento.

Los operadores booleanos encuentran aquellos segmentos en los que la clave especificada es `true` o `false`.

Operadores booleanos

- none: la expresión es verdadera si la palabra clave es verdadera.
- `!`: la expresión es verdadera si la palabra clave es falsa.
- =,`!=`: compara el valor de la palabra clave con la cadena `true` o `false`. Estos operadores actúan igual que los demás, pero son más explícitos.

Example– estado de respuesta es 2XX OK

```
ok
```

Example– el estado de respuesta no es 2XX OK

```
!ok
```

Example– el estado de respuesta no es 2XX OK

```
ok = false
```

Example– el último registro de seguimiento de error enumerado tiene el nombre de error "deserialize"

```
rootcause.fault.entity { last and name = "deserialize" }
```

Example– solicitudes con segmentos remotos donde la cobertura es mayor que 0.7 y el nombre del servicio es "rastros"

```
rootcause.responsetime.entity { remote and coverage > 0.7 and name = "traces" }
```

Example: solicitudes con segmentos inferidos donde el tipo de servicio "AWS:DynamoDB"

```
rootcause.fault.service { inferred and name = traces and type = "AWS::DynamoDB" }
```

Example– solicitudes que tienen un segmento con el nombre "data-plane" como raíz.

```
service("data-plane") {root = true and fault = true}
```

Palabras clave numéricas

Utilice palabras clave numéricas para buscar solicitudes con un tiempo de respuesta, duración o estado de respuesta específico.

Palabras clave numéricas

- `responsetime`: tiempo que tardó el servidor en enviar una respuesta.
- `duration`: duración total de la solicitud, incluidas las llamadas posteriores.
- `http.status`: código de estado de respuesta.
- `index`: posición de un elemento en una lista.
- `coverage`: porcentaje decimal del tiempo de respuesta de una entidad con respecto al tiempo de respuesta del segmento raíz. Aplicable únicamente a las entidades de causa raíz de tiempo de respuesta.

Operadores numéricos

Las palabras clave numéricas usan los operadores de comparación y de igualdad estándar.

- =, != : la palabra clave es igual o no a un valor numérico.
- <,<=, >,>=: la palabra clave es menor o igual a un valor numérico.

Example– el estado de respuesta no es 200 OK

```
http.status != 200
```

Example– solicitud con duración total de 5 a 8 segundos

```
duration >= 5 AND duration <= 8
```

Example– solicitudes que se completaron sin errores en menos de 3 segundos, incluidas todas las llamadas posteriores

```
ok !partial duration <3
```

Example– entidad de lista enumerada que tiene un índice mayor que 5

```
rootcause.fault.service { index > 5 }
```

Example: solicitudes en las que la última entidad tiene una cobertura superior a 0.8

```
rootcause.responsetime.entity { last and coverage > 0.8 }
```

Palabras clave de cadenas

Utilice palabras clave de cadena para encontrar registros de seguimiento con texto específico en los encabezados de solicitud o ID de usuario específicos.

Palabras clave de cadenas

- `http.url`: URL de solicitud.
- `http.method`: método de solicitud.
- `http.useragent`: cadena del agente de usuario de la solicitud.
- `http.clientip`: dirección IP del solicitante.
- `user`: valor del campo de usuario en cualquier segmento incluido en el registro de seguimiento.
- `name`: el nombre de un servicio o excepción.
- `type`: tipo de servicio.
- `message`: mensaje de excepción.

- `availabilityzone`: valor del campo availabilityzone de cualquier segmento incluido en el registro de seguimiento
- `instance.id`: valor del campo de ID de instancia de cualquier segmento incluido en el registro de seguimiento
- `resource.arn`: valor de campo ARN del recurso de cualquier segmento incluido en el registro de seguimiento

Los operadores de cadena permiten encontrar valores iguales a un texto determinado o que contienen dicho texto. Estos valores se deben escribir siempre entre comillas.

Operadores de cadena

- =, `!=`: la palabra clave es igual o no a un valor numérico.
- `CONTAINS`: la palabra clave contiene una cadena concreta.
- `BEGINSWITH` , `ENDSWITH`: la palabra clave comienza o termina con una cadena concreta.

Example– filtro http.url

```
http.url CONTAINS "/api/game/"
```

Para probar si un registro de seguimiento incluye un campo, independientemente de su valor, compruebe si la cadena está vacía.

Example– filtro user

Encuentre todos los registros de seguimiento con ID de usuario.

```
user CONTAINS ""
```

Example– seleccione los registros de seguimiento con una causa raíz de error que incluyan el servicio denominado "Auth"

```
rootcause.fault.service { name = "Auth" }
```

Example– seleccione los registros de seguimiento con una causa raíz de tiempo de respuesta cuyo último servicio tenga un tipo de DynamoDB

```
rootcause.responsetime.service { last and type = "AWS::DynamoDB" }
```

Example– seleccione los registros de seguimiento con una causa raíz de error cuya última excepción tenga el mensaje "access denied for account_id: 1234567890"

```
rootcause.fault.exception { last and message = "Access Denied for account_id:
 1234567890"
```

Palabras clave complejas

Utilice palabras clave complejas para buscar solicitudes basadas en nombre del servicio, nombre de borde o valor de anotación. Para servicios y límites, puede especificar una expresión de filtro adicional que se aplica al servicio o al límite. Para anotaciones, puede filtrar por el valor de una anotación con una clave específica, utilizando operadores booleanos, numéricos o de cadena.

Palabras clave complejas

- `annotation[`*`key`*`]`: valor de anotación con campo *`key`*. Una anotación puede tener un valor booleano, numérico o de cadena, por lo que puede usar cualquiera de estos tipos de operadores de comparación. Puede utilizar esta palabra clave en combinación con la palabra clave `service` o `edge`. Una clave de anotación que contenga puntos debe aparecer entre corchetes ([ ]).
- `edge(`*`source`*`, `*`destination`*`) {`*`filter`*`}`:conexión entre los servicios *`origen`* y *`destino`*. Las llaves opcionales pueden contener una expresión de filtro que se aplica a los segmentos de esta conexión.
- `group.`*`name`* ` / ` `group.`*`arn`*:l valor de la expresión de filtro de un grupo, al que se hace referencia mediante el nombre del grupo o el ARN del grupo.
- `json`: objeto de causa raíz JSON. Consulte Obtener datos de AWS X-Ray para ver los pasos que hay que seguir para crear entidades JSON mediante programación.
- `service(`*`name`*`) {`*`filter`*`}`: servicio denominado *`nombre`*. Las llaves opcionales pueden contener una expresión de filtro que se aplica a los segmentos que creó el servicio.

Utilice la palabra clave service para encontrar los rastros de las solicitudes que se encuentran en un nodo determinado de su mapa de rastros.

Los operadores de palabras clave complejas buscan segmentos en los que se haya establecido o no se haya establecido la clave especificada.

Operadores de palabras clave complejas

- none: la expresión es verdadera si la palabra clave está establecida. Si la palabra clave es de tipo booleano, se evaluará en función del valor booleano.
- `!`: la expresión es verdadera si la palabra clave no está establecida. Si la palabra clave es de tipo booleano, se evaluará en función del valor booleano.
- =, `!=`: compara el valor de la palabra clave.
- `edge(`*`source`*`, `*`destination`*`) {`*`filter`*`}`:conexión entre los servicios *origen* y *destino*. Las llaves opcionales pueden contener una expresión de filtro que se aplica a los segmentos de esta conexión.
- `annotation[`*`key`*`]`: valor de anotación con campo *key*. Una anotación puede tener un valor booleano, numérico o de cadena, por lo que puede usar cualquiera de estos tipos de operadores de comparación. Puede utilizar esta palabra clave en combinación con la palabra clave `service` o `edge`.
- `json`: objeto de causa raíz JSON. Consulte Obtener datos de AWS X-Ray para ver los pasos que hay que seguir para crear entidades JSON mediante programación.

Utilice la palabra clave service para encontrar los rastros de las solicitudes que se encuentran en un nodo determinado de su mapa de rastros.

Example– filtro Service

Solicitudes que incluyen una llamada a `api.example.com` con un error (error de la serie 500).

```
service("api.example.com") { fault }
```

Puede excluir el nombre del servicio para aplicar un filtro de expresión a todos los nodos de su mapa de servicios.

Example– filtro service

Solicitudes que provocan un error en cualquier ubicación de su mapa de rastros.

```
service() { fault }
```

La palabra clave edge aplica una expresión de filtro a una conexión entre dos nodos.

Example– filtro edge

Solicitud en la que el servicio `api.example.com` hizo una llamada a `backend.example.com` que resultó errónea.

```
edge("api.example.com", "backend.example.com") { error }
```

También puede utilizar el operador `!` con las palabras clave service y edge para excluir un servicio o un límite de los resultados de otra expresión de filtro.

Example– filtro service y de solicitud

Solicitud cuya URL comienza por `http://api.example.com/` y contiene `/v2/`, pero no llega al servicio denominado `api.example.com`.

```
http.url BEGINSWITH "http://api.example.com/" AND http.url CONTAINS "/v2/" AND !
service("api.example.com")
```

Example— filtro service y de tiempo de respuesta

Busque rastros en los que esté establecido `http url` y el tiempo de respuesta sea superior a 2 segundos.

```
http.url AND responseTime > 2
```

Para las anotaciones, puede llamar a todos los rastros en los que esté establecido `annotation[`*`key`*`]` o utilizar los operadores de comparación que correspondan al tipo de valor.

Example– anotación con un valor de cadena

Las solicitudes con una anotación denominada `gameid` que tiene el valor de cadena `"817DL6VO"`.

```
annotation[gameid] = "817DL6VO"
```

Example– la anotación está establecida

Solicitudes que tienen establecida una anotación denominada `age`.

```
annotation[age]
```

Example– la anotación no está establecida

Solicitudes que no tienen establecida una anotación denominada `age`.

```
!annotation[age]
```

Example– anotación con un valor numérico

Solicitudes con una edad de anotación con valor numérico mayor que 29.

```
annotation[age] > 29
```

Example– anotación en combinación con service o edge

```
service { annotation[request.id] = "917DL6VO" }
```

```
edge { source.annotation[request.id] = "916DL6VO" }
```

```
edge { destination.annotation[request.id] = "918DL6VO" }
```

Example– grupo con usuario

Solicitudes en las que los rastros cumplan los criterios del filtro de grupo `high_response_time` (p. ej.`responseTime > 3`) y el nombre del usuario sea Alice.

```
group.name = "high_response_time" AND user = "alice"
```

Example– JSON con entidad de causa raíz

Solicitudes con entidades de causa raíz coincidentes.

```
rootcause.json = #[{ "Services": [ { "Name": "GetWeatherData", "EntityPath": [{ "Name":
 "GetWeatherData" }, { "Name": "get_temperature" } ] }, { "Name": "GetTemperature",
 "EntityPath": [ { "Name": "GetTemperature" } ] } ] }]
```

función id

Cuando se proporciona un nombre de servicio a las palabras clave `service` o `edge`, se obtienen resultados de todos los nodos que tienen ese nombre. Si desea filtrar de manera más precisa, puede

utilizar la función `id` para especificar un tipo de servicio, además de un nombre, para diferenciar los nodos con el mismo nombre.

Utilice la función `account.id` para especificar una cuenta concreta para el servicio cuando consulte rastros de varias cuentas de una cuenta de supervisión.

```
id(name: "service-name", type:"service::type", account.id:"account-ID")
```

Puede utilizar la función `id` en lugar de un nombre de servicio en los filtros service y edge.

```
service(id(name: "service-name", type:"service::type")) { filter }
```

```
edge(id(name: "service-one", type:"service::type"), id(name: "service-two",
 type:"service::type")) { filter }
```

Por ejemplo, las funciones AWS Lambda dan como resultado dos nodos en el mapa de rastros: uno para la invocación de la función y otro para el servicio Lambda. Los dos nodos tienen el mismo nombre pero son de diferente tipo. Un filtro service estándar permite encontrar registros de seguimiento para ambos.

Example– filtro service

Solicitudes que incluyen un error en cualquier servicio denominado `random-name`.

```
service("random-name") { error }
```

Utilice la función `id` para limitar la búsqueda a los errores de la propia función, sin incluir los errores del servicio.

Example– filtro service con la función id

Solicitudes que incluyen un error en un servicio denominado `random-name` de tipo `AWS::Lambda::Function`.

```
service(id(name: "random-name", type: "AWS::Lambda::Function")) { error }
```

Si desea buscar nodos por tipo, también puede excluir el nombre por completo.

Example– filtro service con la función id y tipo de servicio

Solicitudes que incluyen un error en un servicio de tipo `AWS::Lambda::Function`.

```
service(id(type: "AWS::Lambda::Function")) { error }
```

Para buscar nodos para una Cuenta de AWS concreta, especifique un ID de cuenta.

Example– filtro service con función id e ID de cuenta

Solicitudes que incluyen un servicio dentro de un identificador de cuenta específico `AWS::Lambda::Function`.

```
service(id(account.id: "account-id"))
```

Rastreo entre cuentas

AWS X-Ray admite la observabilidad entre cuentas, lo que permite al usuario supervisar y solucionar problemas en las aplicaciones que abarcan varias cuentas de una Región de AWS. Puede buscar, visualizar y analizar sin problemas sus métricas, registros y rastros en cualquiera de las cuentas vinculadas, como si estuviera operando en una sola cuenta. Eso le proporciona una vista completa de las solicitudes que se transfieren a varias cuentas. Puede ver los rastros entre cuentas en el mapa de rastros de X-Ray y en las páginas de rastros de la consola de CloudWatch.

Los datos de observabilidad compartidos pueden incluir cualquiera de los siguientes tipos de telemetría:

- Métricas en Amazon CloudWatch
- Grupos de registro de Amazon CloudWatch Logs
- Trazas en AWS X-Ray
- Aplicaciones en Información de aplicaciones de Amazon CloudWatch

Configuración de la observabilidad entre cuentas

Para activar la observabilidad entre cuentas, configure una o más cuentas de supervisión de AWS y vincúlelas a varias cuentas de origen. Una cuenta de supervisión es una Cuenta de AWS central que puede ver los datos de observabilidad generados a partir de las cuentas de origen e interactuar con ellos. Una cuenta de origen es una Cuenta de AWS individual que genera datos de observabilidad para los recursos que residen en ella.

Las cuentas de origen comparten sus datos de observabilidad con cuentas de supervisión. Los rastros de cada cuenta de origen se copian en un máximo de cinco cuentas de supervisión. Las

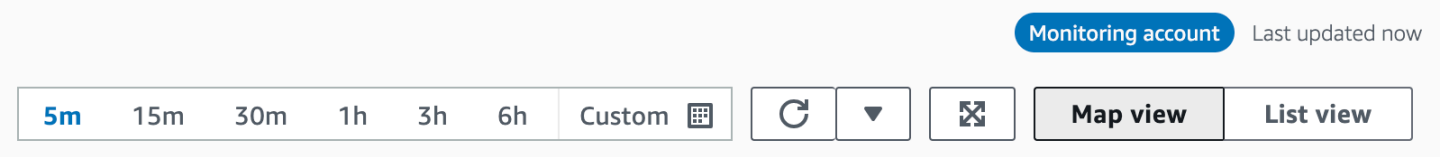
copias de los rastros de las cuentas de origen en la primera cuenta de supervisión son gratuitas. Las copias de rastros enviadas a cuentas de supervisión adicionales se cargan a cada cuenta de origen, según el precio estándar. Para obtener más información consulte Precios de AWS X-Ray y Precios de Amazon CloudWatch.

Para crear enlaces entre las cuentas de supervisión y las de origen, use la consola de CloudWatch o los nuevos comandos de Observability Access Manager de la AWS CLI y la API. Para obtener más información, consulte (Observabilidad entre cuentas de CloudWatch).

> **Note**
>
> Los rastros de X-Ray se cargan en la Cuenta de AWS en la que se reciben. Si una solicitud muestreada abarca varios servicios de más de una Cuenta de AWS, cada cuenta registra un rastro independiente y todos los rastros comparten el mismo ID de rastro. Para obtener más información sobre los precios de la observabilidad entre cuentas, consulte Preciso de AWS X-Ray y Precios de Amazon CloudWatch.

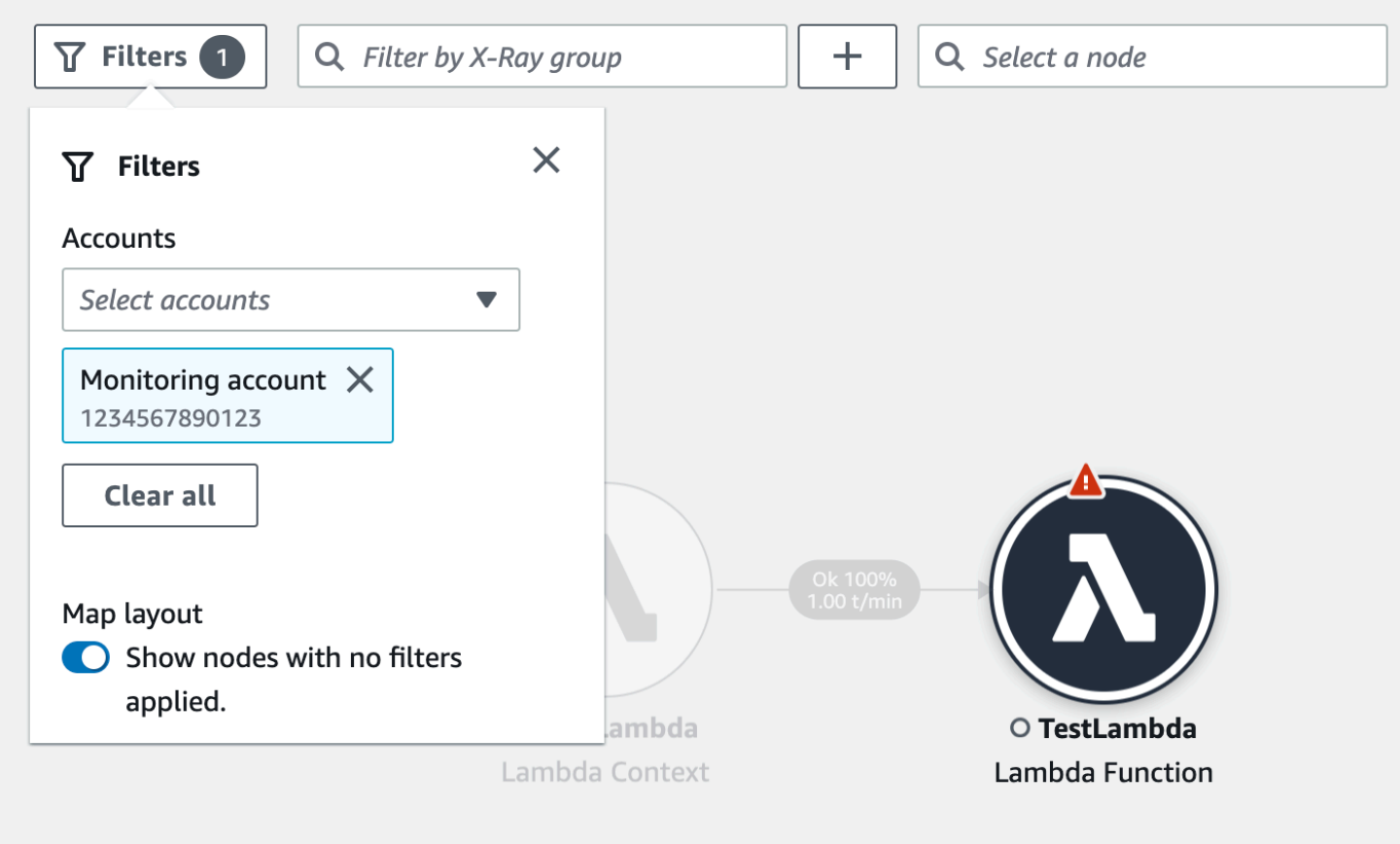
Visualización de rastros entre cuentas

Los rastros entre cuentas se muestran en la cuenta de supervisión. Cada cuenta de origen muestra solo los rastros locales de esa cuenta específica. En las siguientes secciones se presupone que ha iniciado sesión en la cuenta de supervisión y ha abierto la consola de Amazon CloudWatch. Tanto en el mapa de rastros como en la página de rastros, aparece una insignia de cuenta de supervisión en la esquina superior derecha.

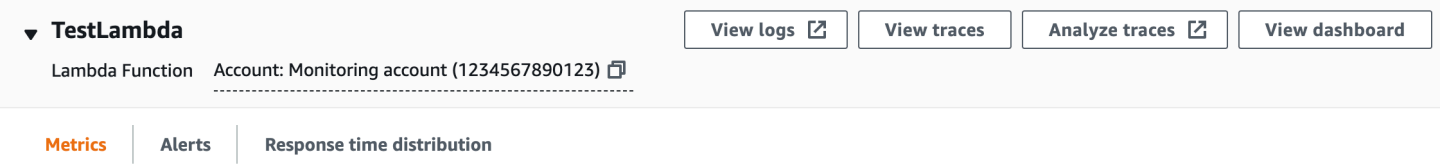


Mapa de rastros

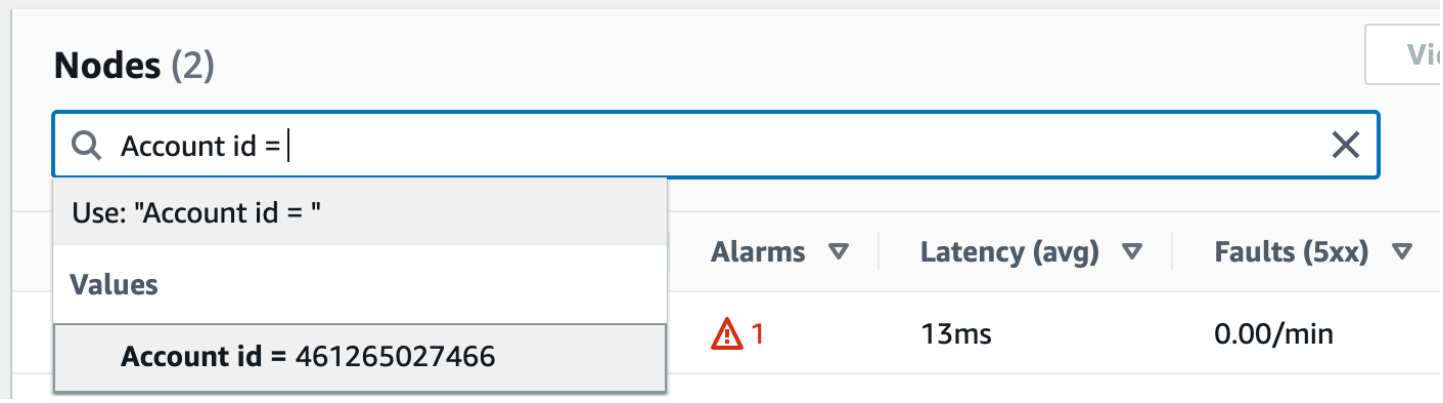
En la consola de CloudWatch, seleccione Mapa de rastros en Rastros de X-Ray, en el panel de navegación izquierdo. De forma predeterminada, el mapa de rastros muestra los nodos de todas las cuentas de origen que envían rastros a la cuenta de supervisión y los nodos de la propia cuenta de supervisión. En el mapa de rastros, elija Filtros en la esquina superior izquierda para filtrar dicho mapa mediante el menú desplegable Cuentas. Una vez que se aplica un filtro de cuenta, los nodos de servicio de las cuentas que no coincidan con el filtro actual aparecen en color gris.



Al elegir un nodo de servicio, el panel de detalles del nodo incluye el identificador de cuenta y la etiqueta del servicio.



En la esquina superior derecha del mapa de rastros, elija Vista de lista para ver una lista de nodos de servicio. La lista de nodos de servicio incluye los servicios de la cuenta de supervisión y todas las cuentas de origen configuradas. Filtre la lista de nodos por Etiqueta de cuenta o por ID de cuenta seleccionándolos en el filtro de Nodos.

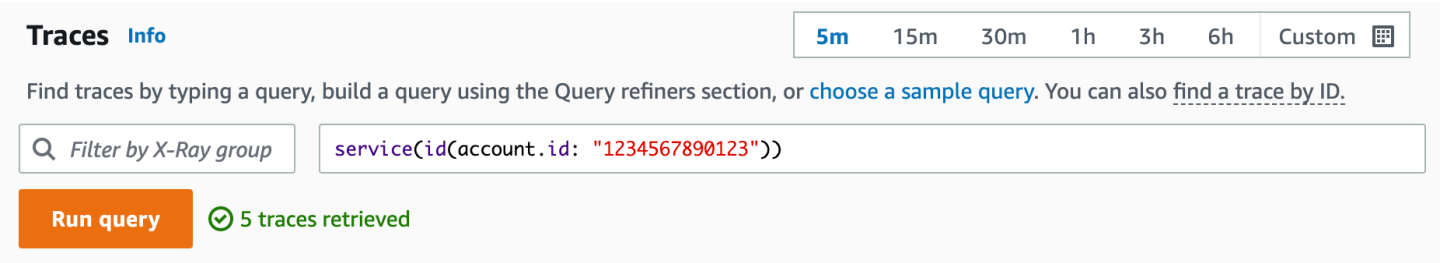


Registros de seguimiento

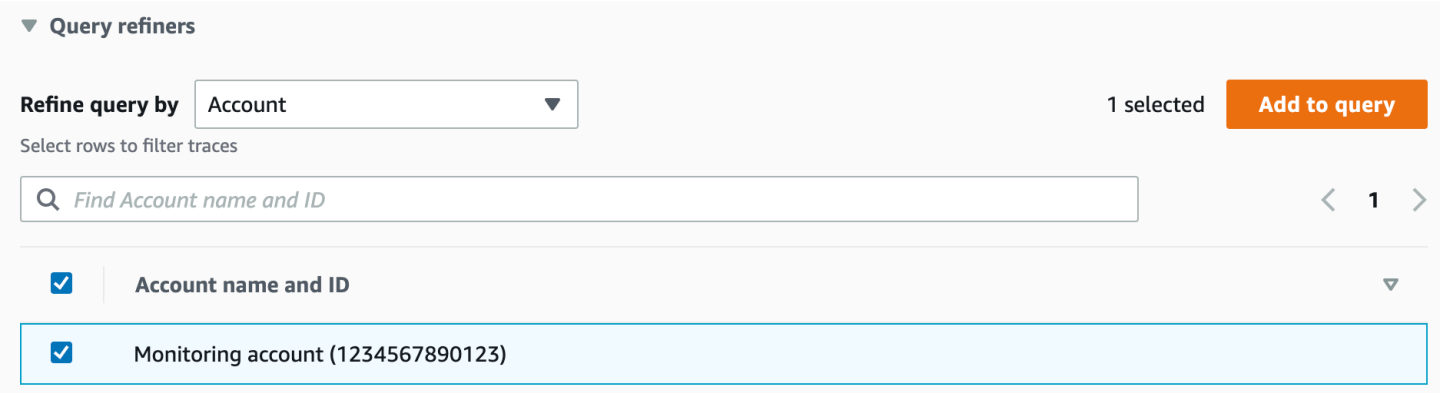
Para ver los detalles de los rastros que abarcan varias cuentas, abra la consola de CloudWatch desde la cuenta de supervisión y elija Rastros en Rastros de X-Ray en el panel de navegación izquierdo. Para abrir esta página también puede seleccionar un nodo en el mapa de rastros de X-Ray y, a continuación, seleccionar Ver rastros en el panel de detalles del nodo.

La página Rastros permite realizar consultas por ID de cuenta. Para empezar, introduzca una consulta que incluya uno o más ID de cuenta. En el siguiente ejemplo se consultan rastros que hayan pasado por el ID de cuenta X o Y:

```
service(id(account.id:"X")) OR service(id(account.id:"Y"))
```



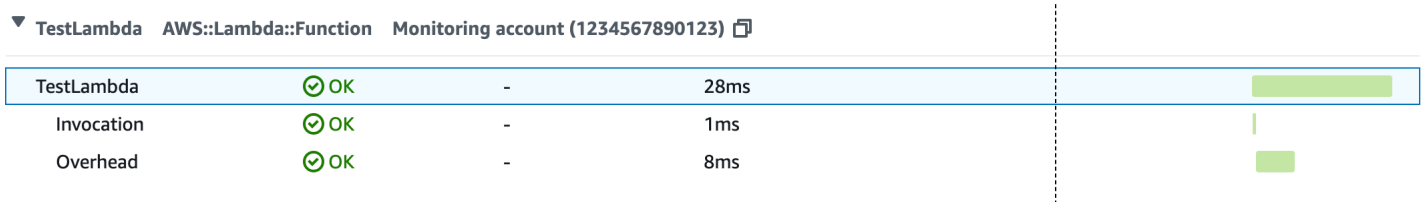
Acote su consulta por Cuenta. Seleccione una o más cuentas de la lista y elija Añadir a la consulta.



Detalles de rastros

Para ver los detalles de una rastro, selecciónelo en la lista Rastros situada en la parte inferior de la página de Rastros. Se muestran los detalles de rastro, incluido un mapa de detalles del rastro con los nodos de servicio de todas las cuentas por las que ha pasado el rastro. Elija un nodo de servicio específico para ver su cuenta correspondiente.

La sección Escala de tiempo de los segmentos muestra los detalles de la cuenta de cada segmento en la escala de tiempo.



Rastreo de aplicaciones basadas en eventos

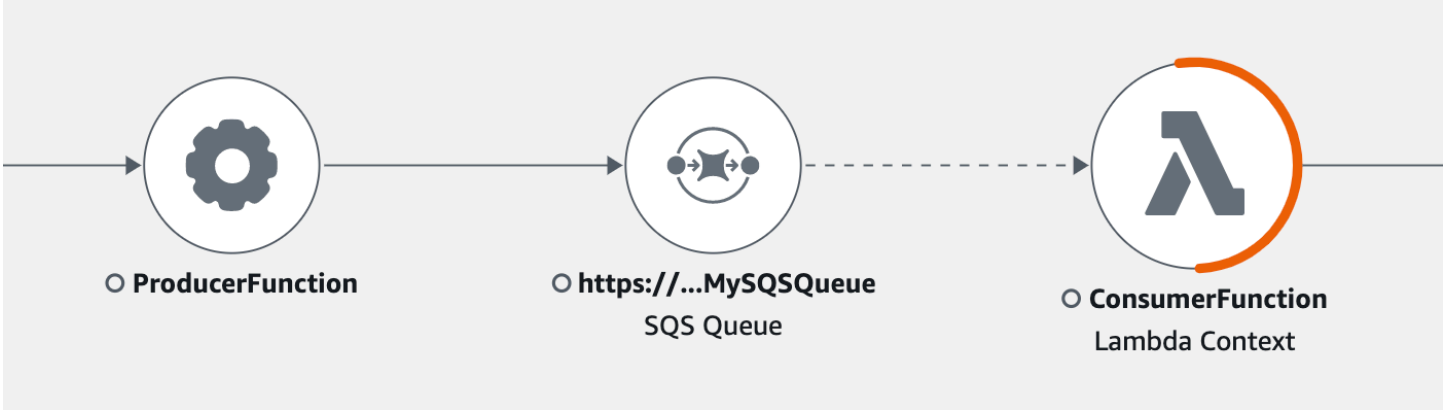
AWS X-Ray admite el rastreo de aplicaciones basadas en eventos mediante Amazon SQS y AWS Lambda. Utilice la consola de CloudWatch para ver una vista conectada de cada solicitud a medida que se pone en cola con Amazon SQS y la procesa una o más funciones de Lambda. Los rastros de los productores de mensajes anteriores se vinculan automáticamente a los rastros de los nodos de consumidor de Lambda posteriores, lo que crea una vista integral de toda la aplicación.

> **Note**
>
> Cada segmento de rastro se puede vincular a un máximo de 20 rastros, mientras que un rastro puede incluir un máximo de 100 enlaces. En algunos escenarios, vincular rastros adicionales puede hacer que se exceda el tamaño máximo de documento de rastro, lo que podría provocar un rastro incompleto. Esto puede suceder, por ejemplo, cuando una función

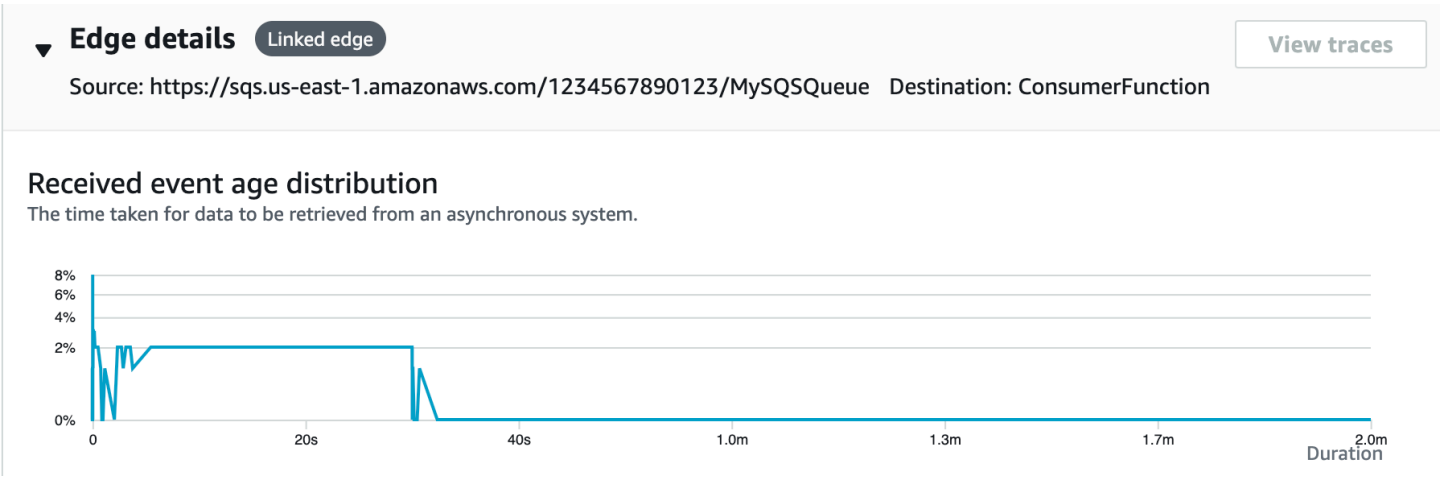
de Lambda con el rastreo habilitado envía muchos mensajes SQS a una cola en una única invocación. Si se produce este problema, hay disponible una solución de mitigación que utiliza los SDK de X-Ray. Consulte el SDK de X-Ray para Java, Node.js, Python, Go o o.NET para obtener más información.

Visualización de rastros vinculados en el mapa de rastros

Utilice la página mapa de rastros de la consola de CloudWatch para ver un mapa de rastros con los rastros de los productores de mensajes que están vinculados a los rastros de los consumidores de Lambda. Estos vínculos se muestran con un borde discontinuo que conecta el nodo de Amazon SQS y los nodos de consumidores de Lambda posteriores.



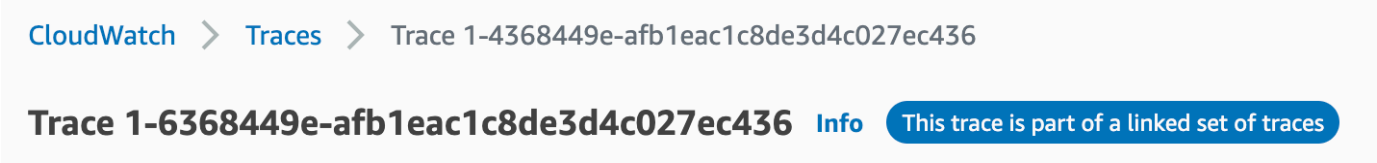
Seleccione un borde discontinuo para mostrar un histograma de antigüedad del evento recibido, que correlaciona la distribución de la antigüedad del evento cuando lo reciben los consumidores. La antigüedad se calcula cada vez que se recibe un evento.



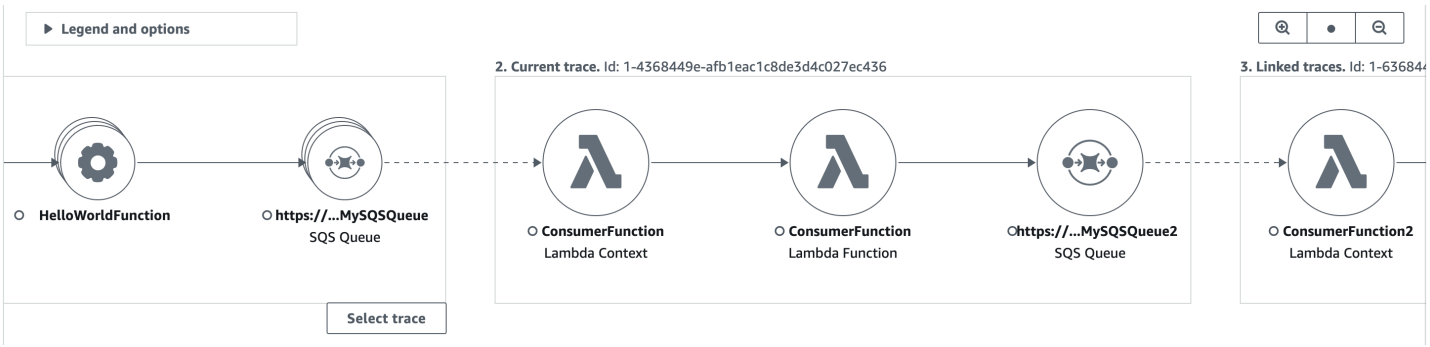
Visualización de detalles de rastro vinculados

Vea los detalles de rastro enviados desde un productor de mensajes, una cola de Amazon SQS o un consumidor de Lambda:

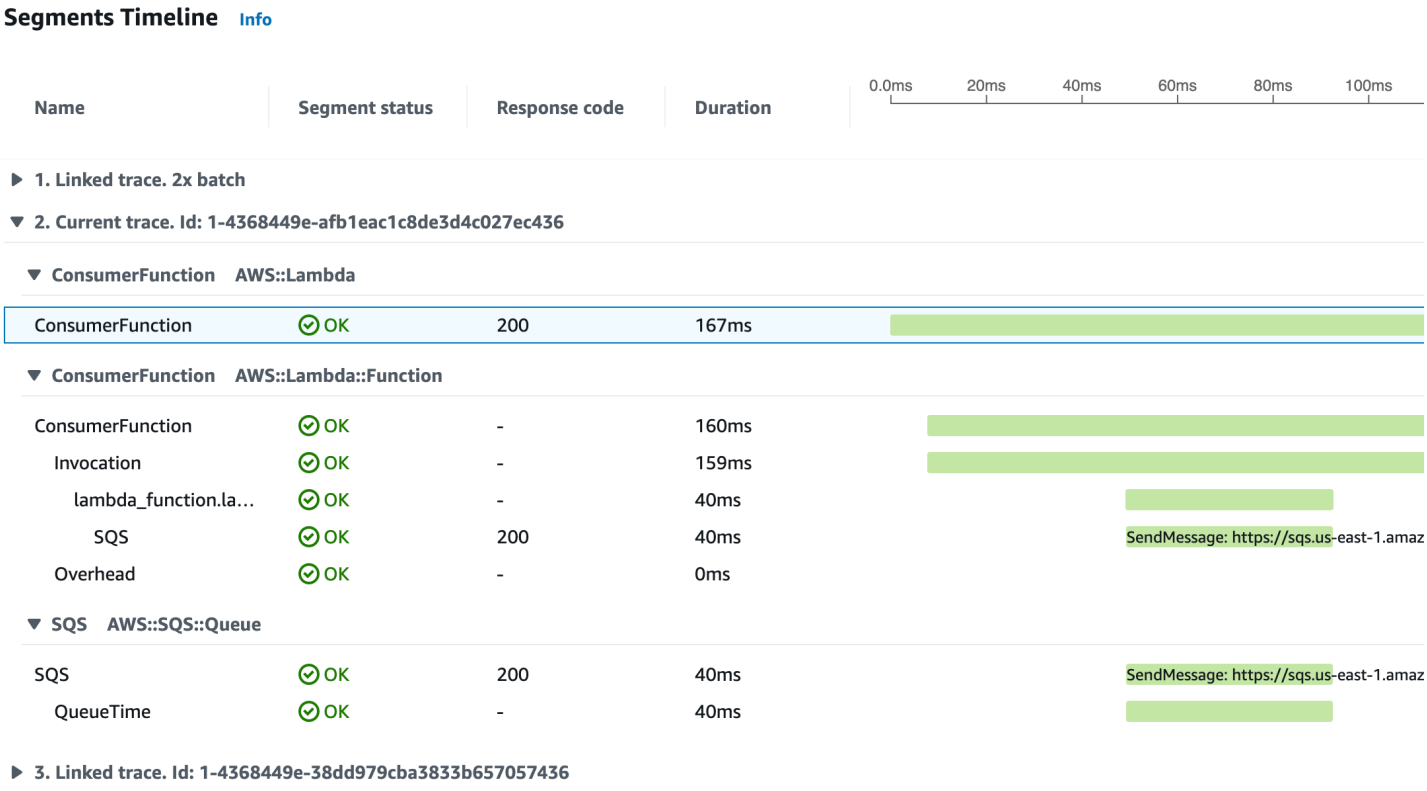
1. Utilice el mapa de rastros para seleccionar un nodo de productor de mensajes, Amazon SQS o consumidor de Lambda.
2. Seleccione Ver rastros en el panel de detalles del nodo para ver una lista de rastros. También puede ir directamente a la página Rastros en la consola de CloudWatch.
3. Elija un rastro específico de la lista para abrir la página de detalles de ese rastro. La página de detalles del rastro muestra un mensaje cuando el rastro seleccionado forma parte de un conjunto de rastros vinculados.



Los datos del mapa de rastros muestra el rastro actual, junto con los rastros vinculados precedentes y posteriores, cada uno de las cuales se encuentra dentro de un recuadro que indica los límites de cada rastro. Si el rastro actualmente seleccionado está vinculado a varios rastros precedentes o posteriores, los nodos que están dentro de los rastros vinculados precedentes o posteriores se apilan y aparece el botón Seleccionar rastro.



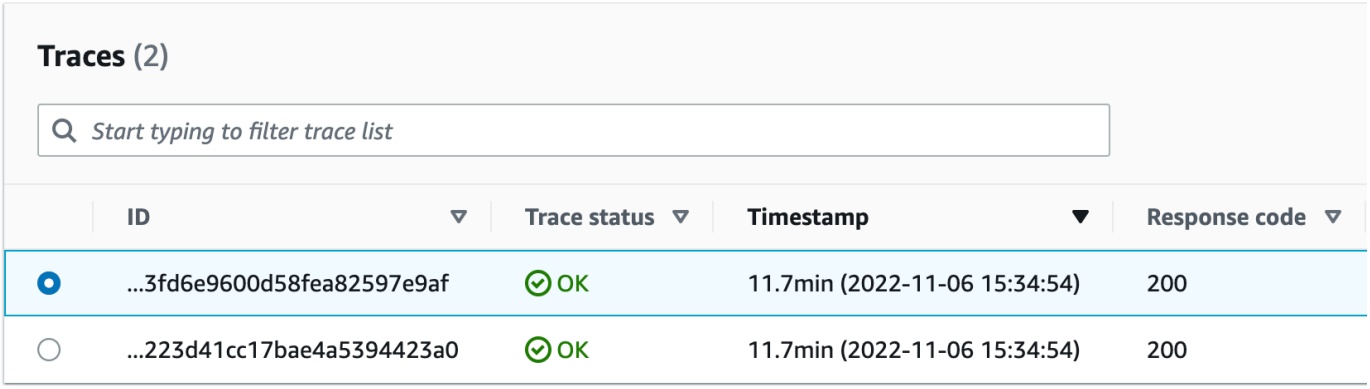
Debajo de los datos del mapa de rastros, se muestra una escala de tiempo de los segmentos de rastros, incluidos los rastros precedentes y posteriores vinculados. Si hay varios rastros precedentes y posteriores vinculados, no se pueden mostrar los detalles de sus segmentos. Para ver los detalles del segmento de una solo rastro dentro de un conjunto de rastros vinculados, seleccione un solo rastro como se describe a continuación.



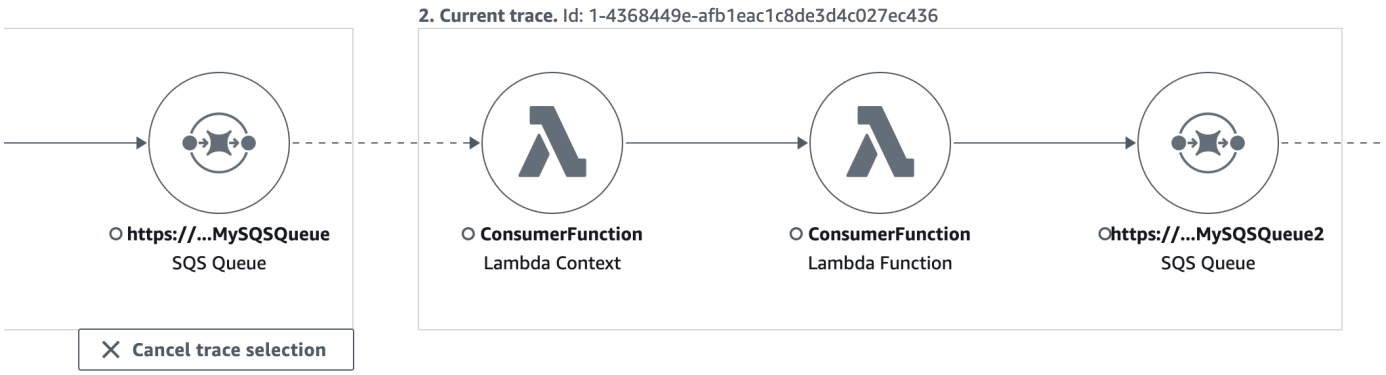
Selección de un solo rastro dentro de un conjunto de rastros vinculados

Filtre un conjunto vinculado de rastros para seleccionar un solo rastro y ver los detalles del segmento en la escala de tiempo.

1. Elija Seleccionar rastro debajo de los rastros vinculados en el mapa de datos del rastro. Aparece una lista de rastros.



2. Seleccione el botón de opción situado junto al rastro para el rastro en el mapa de datos de rastro.
3. Elija Cancelar la selección de rastro para ver todo el conjunto de rastros vinculados.



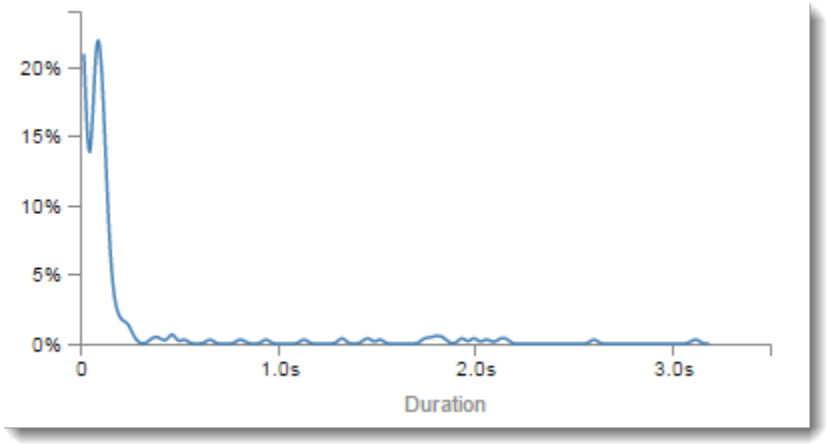
Uso de histogramas de latencia

Al seleccionar un nodo o un borde en un mapa de rastros de AWS X-Ray, la consola de X-Ray muestra un histograma de distribución de latencias.

Latencia

La latencia es el tiempo que transcurre entre el momento de iniciar una solicitud y el momento de completarla. Los histogramas muestran una distribución de las latencias. Muestran la duración en el eje x y el porcentaje de solicitudes de cada duración en el eje y.

Este histograma muestra un servicio que completa la mayor parte de las solicitudes en menos de 300 ms. Un pequeño porcentaje de solicitudes tarda hasta 2 segundos y unos cuantos casos atípicos tardan más tiempo.



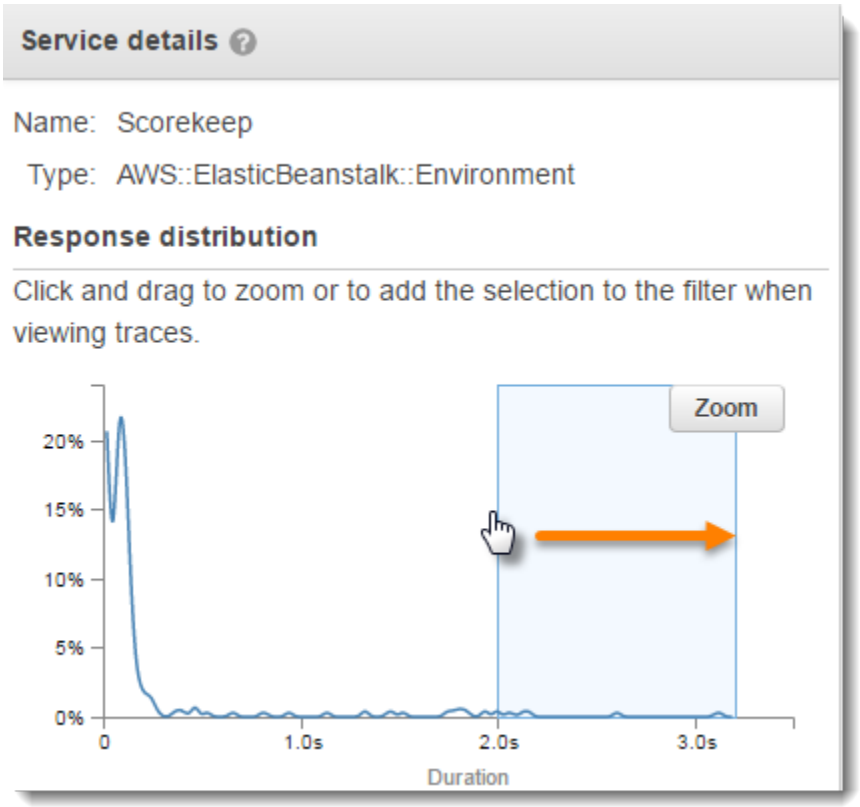
Interpretación de los detalles del servicio

Los histogramas de servicio y los histogramas de límites constituyen una representación visual de latencia desde el punto de vista de un servicio o de un solicitante.

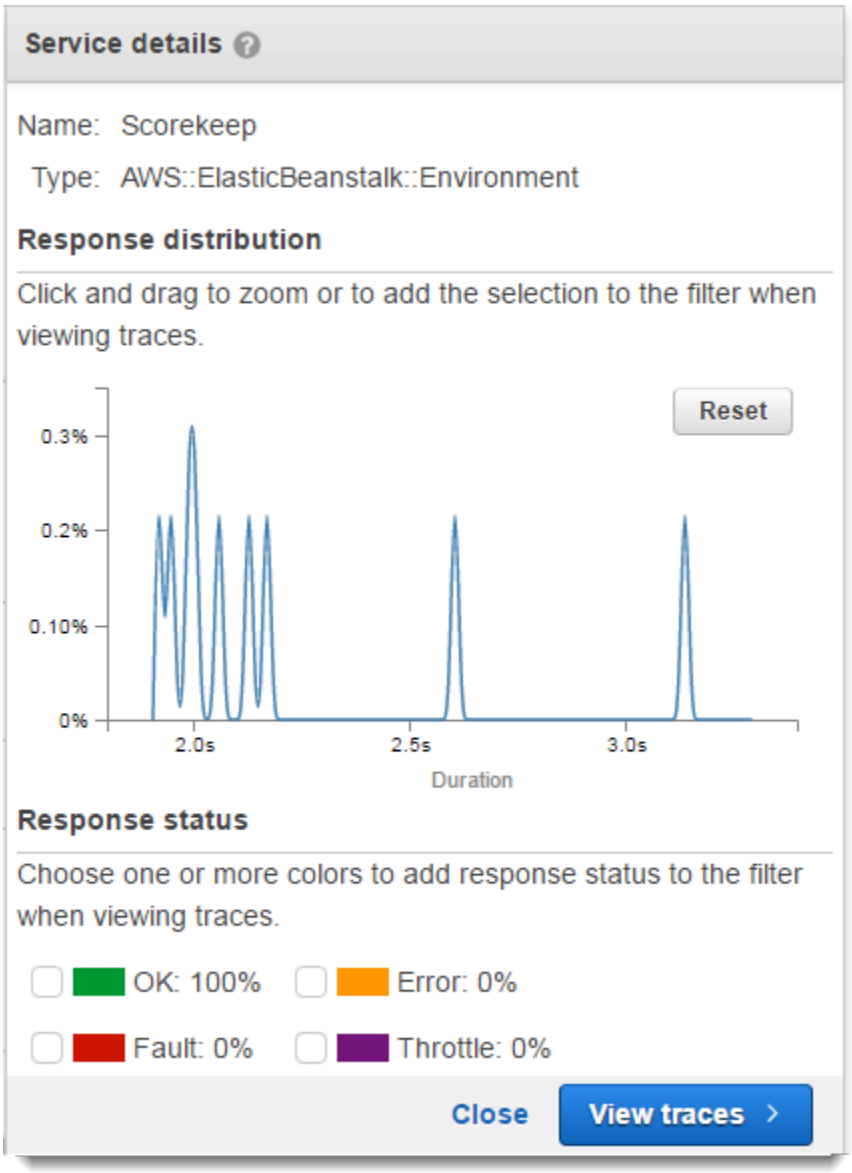
- Elija un nodo de servicio haciendo clic en el círculo. X-Ray muestra un histograma de las solicitudes que atiende el servicio. Las latencias son las que registra el servicio y no incluyen ninguna latencia de red entre el servicio y el solicitante.
- Elija un borde haciendo clic en la línea o en la punta de flecha del borde entre dos servicios. X-Ray muestra un histograma de las solicitudes del solicitante atendidas por el servicio posterior. Las latencias son las que ha registrado el solicitante e incluyen la latencia de la conexión de red entre los dos servicios.

Para interpretar el panel de histograma Service details, puede buscar los valores que más difieran de la mayoría de valores del histograma. Estos valores atípicos pueden aparecer como picos o puntas en el histograma y puede analizar los rastros de un área específica para averiguar lo que sucede.

Para ver los rastros filtrados por latencia, seleccione un intervalo en la histograma. Haga clic donde desee iniciar la selección y arrastre de izquierda a derecha para resaltar el intervalo de latencias que quiera incluir en el filtro de rastros.



Una vez seleccionado el intervalo, puede elegir Zoom (Zoom) para ver solo esa parte del histograma y retocar la selección.



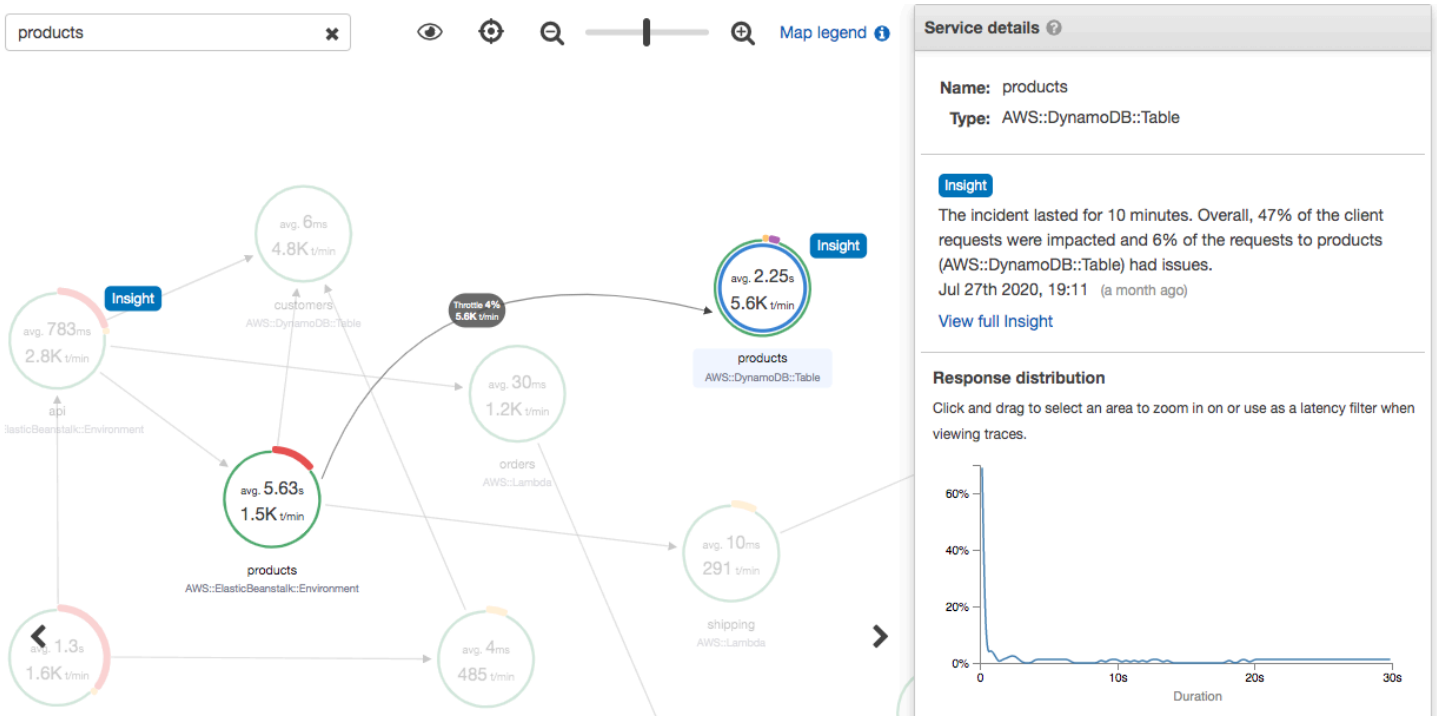
Una vez establecido el enfoque en el área que desea, elija View traces (Ver rastros).

Uso de información en X-Ray

AWS X-Ray analiza continuamente los datos de rastro de su cuenta para identificar problemas emergentes en sus aplicaciones. Cuando las tasas de fallo superan el rango esperado, se crea una información que registra el problema y hace un seguimiento de su impacto hasta que se resuelve. Con la información, puede:

- Identificar en qué parte de su aplicación se ha producido el problema, la causa raíz del problema y el impacto asociado. El análisis de impacto que proporciona la información le permite deducir la gravedad y la prioridad de un problema.
- Reciba notificaciones a medida que el problema cambie con el tiempo. Puede integrar las notificaciones de información en su solución de supervisión y alertas mediante Amazon EventBridge. Esta integración le permite enviar correos electrónicos o alertas automatizados en función de la gravedad del problema.

La consola de X-Ray identifica los nodos con incidentes en curso en el mapa de rastros. Para ver un resumen de la información, elija el nodo afectado. También puede ver y filtrar la seleccionando Información en el panel de navegación de la izquierda.



X-Ray crea información cuando detecta una anomalía en uno o más nodos del mapa de servicio. El servicio utiliza modelos estadísticos para predecir las tasas de fallo que cabe esperar de los servicios de su aplicación. En el ejemplo anterior, la anomalía es un aumento de los fallos de AWS Elastic Beanstalk. El servidor de Elastic Beanstalk agotó varias veces los tiempos de espera de llamadas a la API, lo que provocó una anomalía en los nodos posteriores.

Habilitación de la información en la consola de X-Ray

La información debe estar habilitada para cada grupo con el que desee utilizar las características de información. Puede activar la información desde la página Grupos.

1. Abra la consola de X-Ray.
2. Seleccione un grupo existente o cree uno nuevo seleccionando Crear grupo y, a continuación, seleccione Habilitar Información. Para obtener más información acerca de la configuración de grupos en la consola de X-Ray, consulte Configuración de grupos.
3. En el panel de navegación de la izquierda, elija Información y, luego, elija la información que desee ver.

> **Note**
>
> X-Ray utiliza las operaciones de las API de GetInsightSummaries, GetInsight, GetInsightEvents y GetInsightImpactGraph API para recuperar datos de la información. Para obtener más información, consulte Cómo funciona AWS X-Ray con IAM.

Habilitación de notificaciones de la información

Con las notificaciones de información, se crea una notificación para cada evento de información, por ejemplo, cuando se crea una información, cambia significativamente o se cierra. Los clientes pueden recibir estas notificaciones a través de eventos de Amazon EventBridge y utilizar reglas condicionales para realizar acciones como la notificación de SNS, la invocación a Lambda, la publicación de mensajes en una cola de SQS o cualquiera de los destinos compatibles con EventBridge. Las notificaciones de información se emiten en la medida de lo posible, pero no están garantizadas. Para obtener más información sobre destinos, consulte Destinos de Amazon EventBridge.

En la página Grupos puede habilitar las notificaciones de información para cualquier grupo que tenga información habilitada.

Habilitación de notificaciones para un grupo de X-Ray

1. Abra la consola de X-Ray.

2. Seleccione un grupo existente o cree uno nuevo seleccionando Crear grupo, asegúrese de que Habilitar información esté seleccionado y, a continuación, seleccione Habilitar Información. Para obtener más información acerca de la configuración de grupos en la consola de X-Ray, consulte Configuración de grupos.

Configuración de las reglas condicionales de Amazon EventBridge

1. Abra la consola de Amazon EventBridge.
2. Vaya a Reglas en la barra de navegación izquierda y elija Crear regla.
3. Escriba un nombre y una descripción para la regla.
4. Elija Patrón de eventos y, a continuación, Patrón personalizado. Proporcione un patrón que contenga `"source": [ "aws.xray" ]` y `"detail-type": [ "AWS X-Ray Insight Update" ]`. A continuación se muestran algunos ejemplos de posibles patrones.
   - Patrón de eventos que coincide con todos los eventos entrantes de información de X-Ray:

     ```
     {
     "source": [ "aws.xray" ],
     "detail-type": [ "AWS X-Ray Insight Update" ]
     }
     ```

   - Patrón de eventos que coincide con un valor especificado de **`state`** y **`category`**:

     ```
     {
     "source": [ "aws.xray" ],
     "detail-type": [ "AWS X-Ray Insight Update" ],
     "detail": {
            "State": [ "ACTIVE" ],
            "Category": [ "FAULT" ]
       }
     }
     ```

5. Seleccione y configure los destinos que desee invocar cuando un evento cumpla esta regla.
6. (Opcional) Proporcione etiquetas para identificar y seleccionar esta regla con mayor facilidad.
7. Seleccione Crear.

> **Note**
>
> Las notificaciones de información de X-Ray envían eventos a Amazon EventBridge, que actualmente no admite claves administradas por el cliente. Para obtener más información, consulte Protección de los datos en AWS X-Ray.

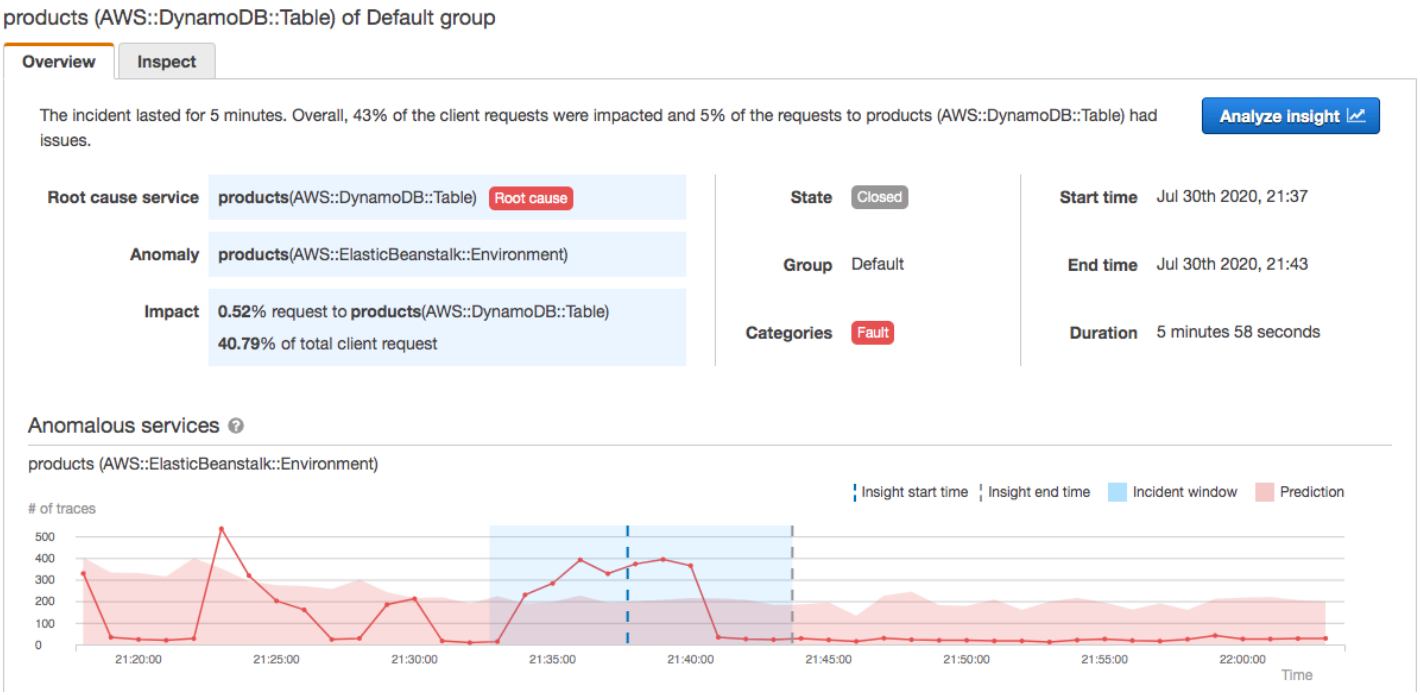
Visión general acerca de la información

En la página de información general de una información se intenta responder a tres preguntas clave:

- ¿Cuál es el problema de fondo?
- ¿Cuál es la causa raíz?
- ¿Cuál es el impacto?

La sección Servicios anómalos muestra una escala de tiempo para cada servicio que ilustra el cambio en las tasas de fallo durante el incidente. La escala de tiempo muestra el número de rastros con fallos superpuestos en una banda continua que indica el número esperado de fallos en función de la cantidad de tráfico registrada. La duración de la información se visualiza en la ventana de incidentes. La ventana de incidentes comienza cuando X-Ray observa que la métrica se vuelve anómala y persiste mientras la información está activa.

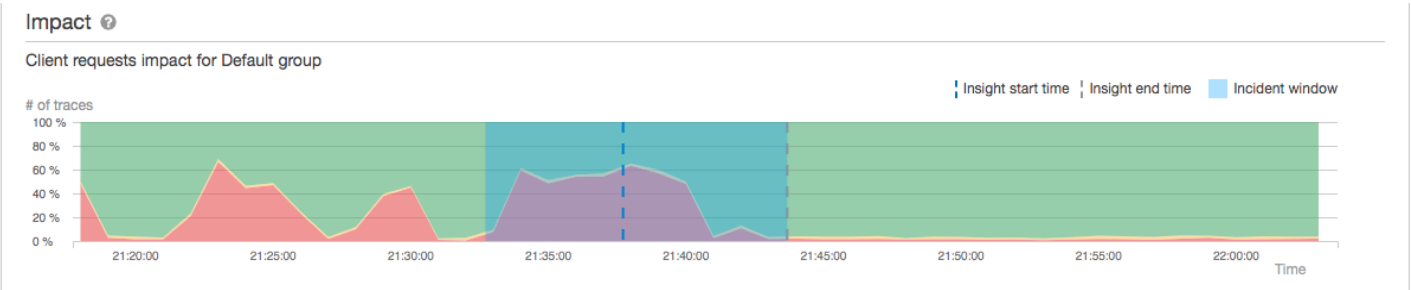
En el siguiente ejemplo se muestra un aumento en el número de fallos, lo que ha provocado un incidente:



La sección Causa raíz muestra un mapa de rastros centrado en el servicio de la causa raíz y la ruta afectada. Puede ocultar los nodos no afectados seleccionando el icono en forma de ojo en la parte superior derecha del mapa de la causa raíz. El servicio de causa raíz es el nodo posterior más alejado en el que X-Ray identificó una anomalía. Puede representar un servicio que usted instrumentó o un servicio externo al que su servicio llamó con un cliente instrumentado. Por ejemplo, si llama a Amazon DynamoDB con un cliente instrumentado del SDK de AWS, un aumento del número de falos de DynamoDB da como resultado una información con DynamoDB como causa principal.

Para investigar más a fondo la causa raíz, seleccione Ver los detalles de la causa raíz en el gráfico de la causa raíz. Puede usar la página Análisis para investigar la causa raíz y los mensajes relacionados con ella. Para obtener más información, consulte Interacción con la consola de Analytics.

Los fallos que continúan hacia arriba en el mapa pueden afectar a varios nodos y provocar múltiples anomalías. Si un fallo se transfiere hasta el usuario que realizó la solicitud, el resultado es un fallo de cliente. Se trata de un fallo en el nodo raíz del mapa de rastros. El gráfico Impacto proporciona una escala de tiempo de la experiencia del cliente para todo el grupo. Esta experiencia se calcula en función de los porcentajes de los siguientes estados: fallo, error, limitación y bien.



En este ejemplo se muestra un aumento en el número de rastros con un fallo en el nodo raíz durante el momento de un incidente. Los incidentes en los servicios posteriores no siempre se corresponden con un aumento de los errores de cliente.

Al elegir Analizar información, se abre la consola de X-Ray Analytics en una ventana en la que se puede profundizar en el conjunto de rastros que han generado la información. Para obtener más información, consulte Interacción con la consola de Analytics.

Descripción del impacto

AWS X-Ray mide el impacto causado por un problema en curso como parte de la generación de información y notificaciones. El impacto se mide de dos formas:

- Impacto en el grupo de X-Ray
- Impacto en el servicio de la causa raíz

Este impacto viene determinado por el porcentaje de solicitudes que fallan o que provocan un error en un período de tiempo determinado. Este análisis de impacto le permite determinar la gravedad y la prioridad del problema en función de su situación particular. Este impacto está disponible como parte de la experiencia con la consola, además de las notificaciones de información.

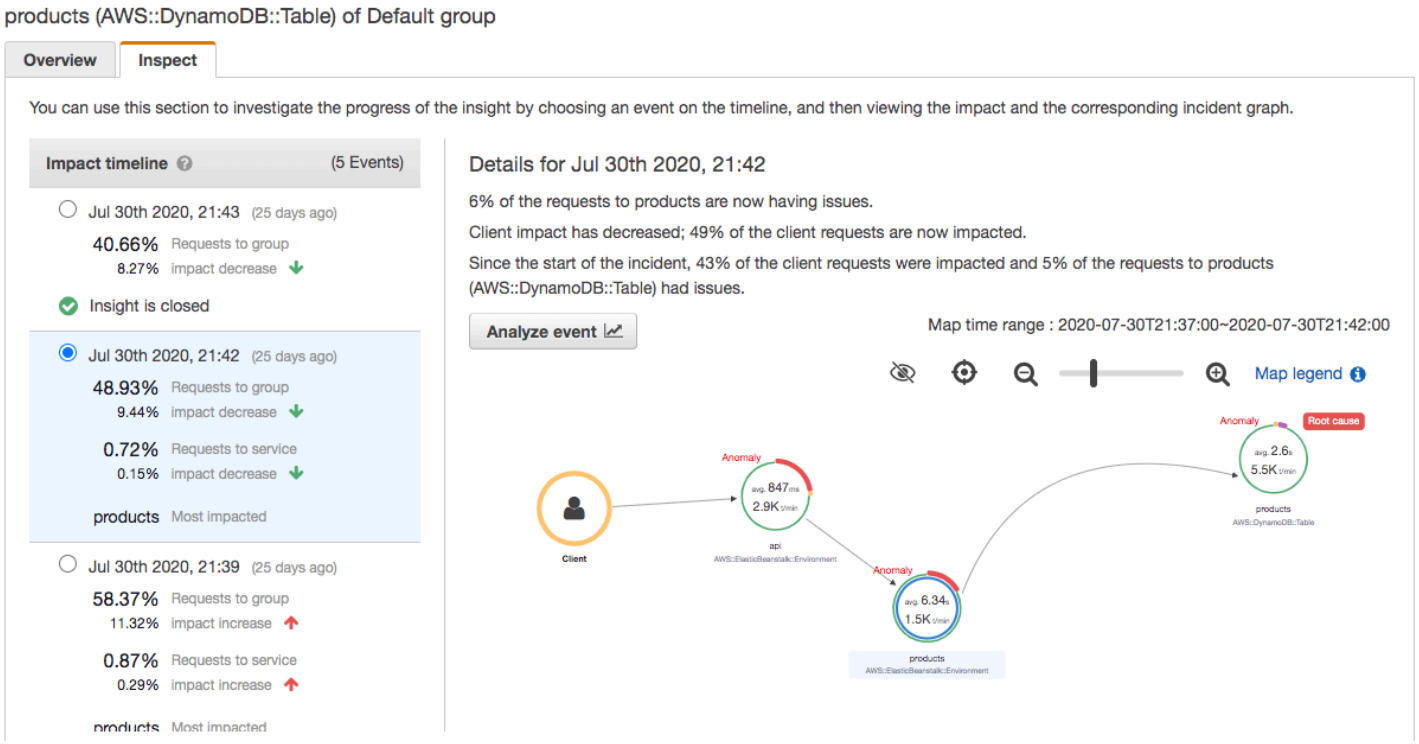
Desduplicación

La información de AWS X-Ray desduplica los problemas en varios microservicios. Utiliza la detección de anomalías para determinar el servicio que es la causa principal de un problema, determina si otros servicios relacionados con él presentan un comportamiento anómalo debido a la misma causa raíz y registra el resultado como una sola información.

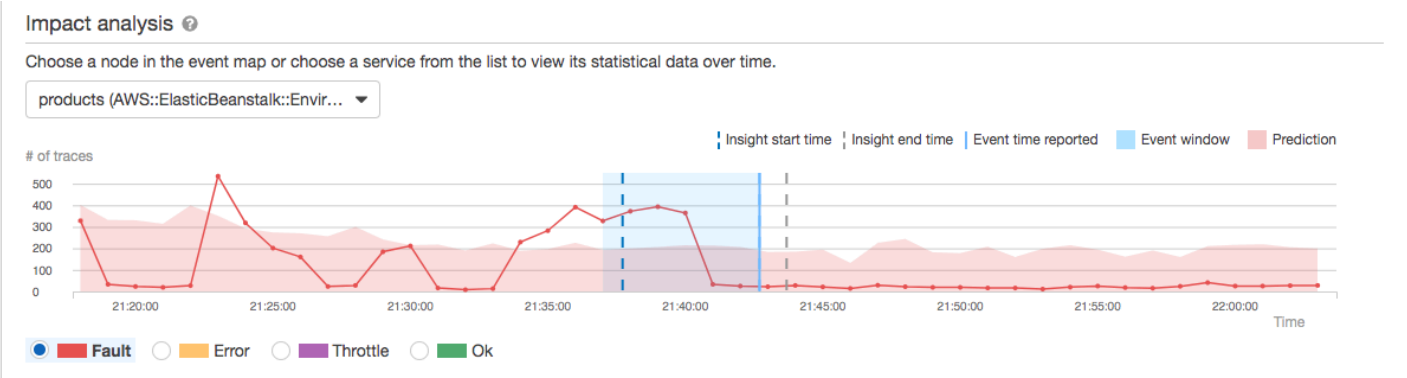
Revisión del progreso de una información

X-Ray reevalúa la información periódicamente hasta que se resuelve y registra cada cambio intermedio notable en forma de notificación, la cual se puede enviar como evento de Amazon EventBridge. Esto le permite crear procesos y flujos de trabajo para determinar cómo ha cambiado el problema a lo largo del tiempo y tomar las medidas adecuadas, como enviar un correo electrónico o integrarse en un sistema de alertas mediante EventBridge.

Puede revisar los eventos de los incidentes en Plazo del impacto, en la página Inspeccionar. De forma predeterminada, la escala de tiempo muestra el servicio más afectado hasta que el usuario elige un servicio diferente.



Para ver un mapa y gráficos de rastros de un evento, elíjalo en el plazo de tiempo del impacto. El mapa de rastros muestra los servicios de su aplicación que se ven afectados por el incidente. En Análisis del impacto, los gráficos muestran las líneas de tiempo de los fallos correspondientes al nodo seleccionado y los clientes del grupo.



Para analizar en profundidad los rastros implicados en un incidente, elija Analizar evento en la página Inspeccionar. Puede utilizar la página Análisis para acotar la lista de rastros e identificar a los usuarios afectados. Para obtener más información, consulte Interacción con la consola de Analytics.

Interacción con la consola de Analytics

La consola de AWS X-Ray Analytics es una herramienta interactiva para interpretar rápidamente los datos de rastreo con el fin de conocer el rendimiento de la aplicación y sus servicios subyacentes. La consola le permite explorar, analizar y visualizar los registros de seguimiento a través de gráficos de tiempo de respuesta y series temporales.

Al realizar selecciones en la consola de Analytics, la consola crea filtros para reflejar el subconjunto seleccionado de registros de seguimiento. Puede acotar el conjunto de datos con filtros cada vez más detallados haciendo clic en los gráficos y los paneles de métricas y campos asociados al conjunto de registros de seguimiento actual.

Temas

- Características de la consola
- Distribución del tiempo de respuesta
- Actividad de series temporales
- Ejemplos de flujo de trabajo
- Observar errores en el gráfico de servicio
- Identificar los picos de tiempo de respuesta
- Consultar todos los registros de seguimiento marcados con un código de estado

- Consultar todos los elementos de un subgrupo y asociados a un usuario
- Comparar dos conjuntos de registros de seguimiento con criterios diferentes
- Identificar un registro de seguimiento de su interés y ver sus detalles

Características de la consola

La consola de X-Ray Analytics utiliza las siguientes características principales para agrupar, filtrar, comparar y cuantificar los datos de rastro.

Características

Característica	Descripción
Grupos	El grupo seleccionado inicial es `Default`. Para cambiar el grupo recuperado, seleccione un grupo diferente en el menú que se encuentra a la derecha de la barra de búsqueda principal de expresiones de filtro. Para obtener más información sobre los grupos, consulte Uso de expresiones de filtro con grupos.
Retrieved traces (Registros de seguimiento recuperados	De forma predeterminada, la consola de Analytics genera gráficos basados en todos los registros de seguimiento del grupo seleccion ado. Los registros de seguimiento recuperados representan todos los registros de seguimien to del conjunto de trabajo. Puede ver el número de registros de seguimiento en este icono. Las expresiones de filtro que se aplican a la barra de búsqueda principal acotan y actualizan los registros de seguimiento recuperados.
Mostrar y ocultar de gráficos	Un control de alternancia para comparar el grupo activo con los registros de seguimiento recuperados. Para comparar los datos relativos al grupo con los filtros activos, elija Show in charts (Mostrar en gráficos). Para eliminar esta

Característica	Descripción
	vista de los gráficos, elija Hide from charts (Ocultar de gráficos).
Filtered trace set A (Conjunto de registros de seguimiento filtrados A	A través de interacciones con los gráficos y las tablas, aplique filtros para crear los criterios del conjunto de rastros filtrados A. Cuando se aplican los filtros, el número de rastros aplicables y el porcentaje de rastros que se recupera se calculan en este icono. Los filtros se rellenan como etiquetas dentro del icono Filtered trace set A (Conjunto de registros de seguimiento A filtrados) y también se pueden eliminar del icono.
Refine (Acotar	Esta función actualiza el conjunto de registros de seguimiento recuperados en función de los filtros aplicados al conjunto de registros de seguimiento A. Al reajustar el conjunto de registros de seguimiento recuperado se actualiza el conjunto de trabajo de todos los registros de seguimiento recuperados en función de los filtros del conjunto de registros de seguimiento A. El conjunto de trabajo de registros de seguimiento recuperados es un subconjunto muestreado de todos los registros de seguimiento del grupo.

Característica	Descripción
Filtered trace set B (Conjunto de registros de seguimiento filtrados B	Cuando se crea, el conjunto de rastros filtrados B es una copia del conjunto de rastros filtrados A. Para comparar los dos conjuntos de rastros, seleccione nuevos filtros, que se aplicarán al conjunto de rastros B, mientras que el conjunto A permanece fijo. Cuando se aplican los filtros, se calcula el número de registros de seguimien to aplicables y el porcentaje de registros de seguimiento del total recuperado en este icono. Los filtros se rellenan como etiquetas dentro del icono Filtered trace set B (Conjunto de registros de seguimiento B) y también se pueden eliminar del icono.
Response Time Root Cause Entity Paths (Rutas de entidad de causa raíz de tiempo de respuesta	Una tabla de rutas de entidad registradas. X-ray determina qué ruta del rastro del usuario es la causa más probable del tiempo de respuesta . El formato indica una jerarquía de entidades detectadas, que termina en una causa raíz de tiempo de respuesta. Utilice estas filas para filtrar errores de tiempo de respuesta recurrentes. Para obtener más información sobre cómo personalizar un filtro de causa raíz y obtener los datos a través de la API, consulte Recuperación y reajuste de análisis de causa raíz.
Delta (�)	Una columna que se agrega a las tablas de métricas cuando los conjuntos de registros de seguimiento A y B están activos. La columna Delta calcula la diferencia porcentual de los registros de seguimiento entre el conjunto de registros de seguimiento A y el conjunto de registros de seguimiento B.

Distribución del tiempo de respuesta

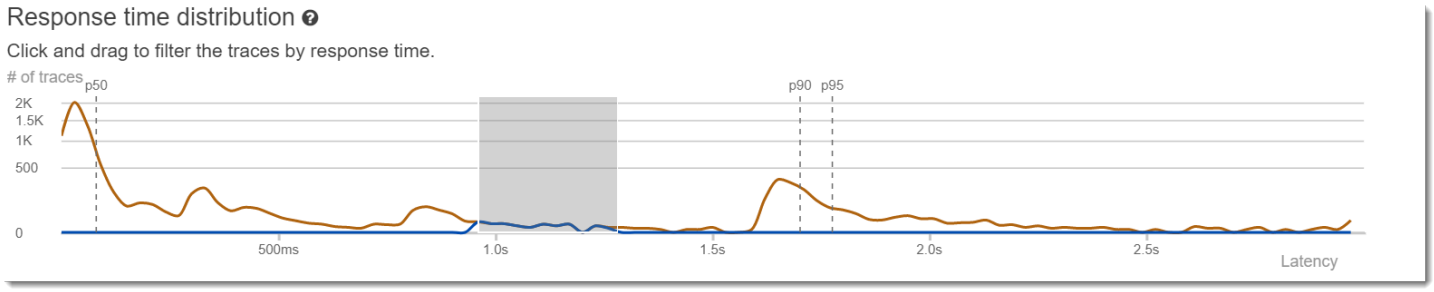
La consola de X-Ray Analytics genera dos gráficos principales que ayudan al usuario a visualizar rastros: Distribución del tiempo de respuesta y Actividad de series temporales. En esta sección y las siguientes se proporcionan ejemplos de cada uno y se explican los aspectos básicos de cómo leer los gráficos.

A continuación se indican los colores asociados al gráfico de líneas de tiempo de respuesta (el gráfico de series temporales utiliza la misma combinación de colores):

- Todos los rastros del grupo: gris
- Rastros recuperados: naranja
- Conjunto de rastros filtrados A: verde
- Conjunto de rastros filtrados B: azul

Example– Distribución del tiempo de respuesta

La distribución del tiempo de respuesta es un gráfico que muestra el número de registros de seguimiento con un tiempo de respuesta determinado. Haga clic y arrastre para realizar selecciones en la distribución del tiempo de respuesta. De esta forma, se selecciona y se crea un filtro del conjunto de registros de seguimiento de trabajo llamado `responseTime` para todos los registros de seguimiento de un tiempo de respuesta específico.

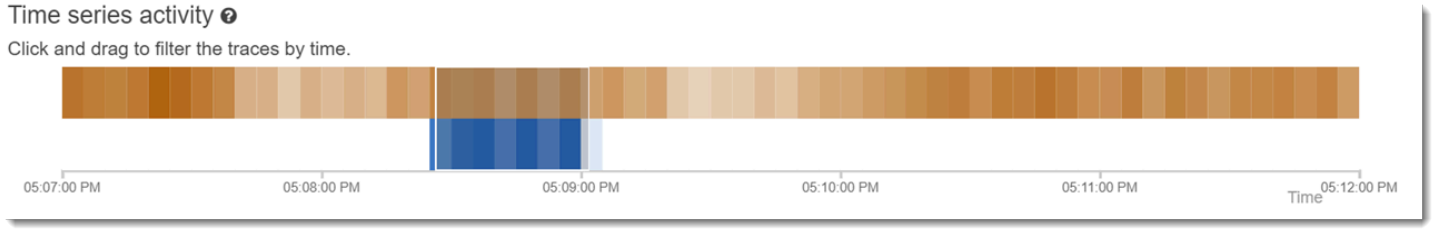


Actividad de series temporales

El gráfico de actividad de series temporales muestra el número de registros de seguimiento en un período de tiempo determinado. Los indicadores de color reflejan los colores del gráfico de líneas de la distribución del tiempo de respuesta. Cuanto más oscuro y opaco es el bloque de color de las series de actividad, más registros de seguimiento se representan en el momento especificado.

Example– Actividad de series temporales

Haga clic y arrastre para realizar selecciones dentro del gráfico de actividad de series temporales. De esta forma, se selecciona y se crea un filtro llamado `timerange` en el conjunto de registros de seguimiento de trabajo para todos los registros de seguimiento en un período de tiempo específico.



Ejemplos de flujo de trabajo

Los siguientes ejemplos muestran casos de uso comunes de la consola de X-Ray Analytics. Cada ejemplo muestra una función clave de la experiencia de la consola. En su conjunto, los ejemplos siguen un flujo de trabajo de solución de problemas básico. Los pasos describen cómo detectar primero los nodos en mal estado y, a continuación, cómo interactuar con la consola de Analytics para generar automáticamente consultas comparativas. Una vez que haya reducido el alcance a través de las consultas, verá los detalles de los registros de seguimiento de interés para determinar qué está dañando el estado del servicio.

Observar errores en el gráfico de servicio

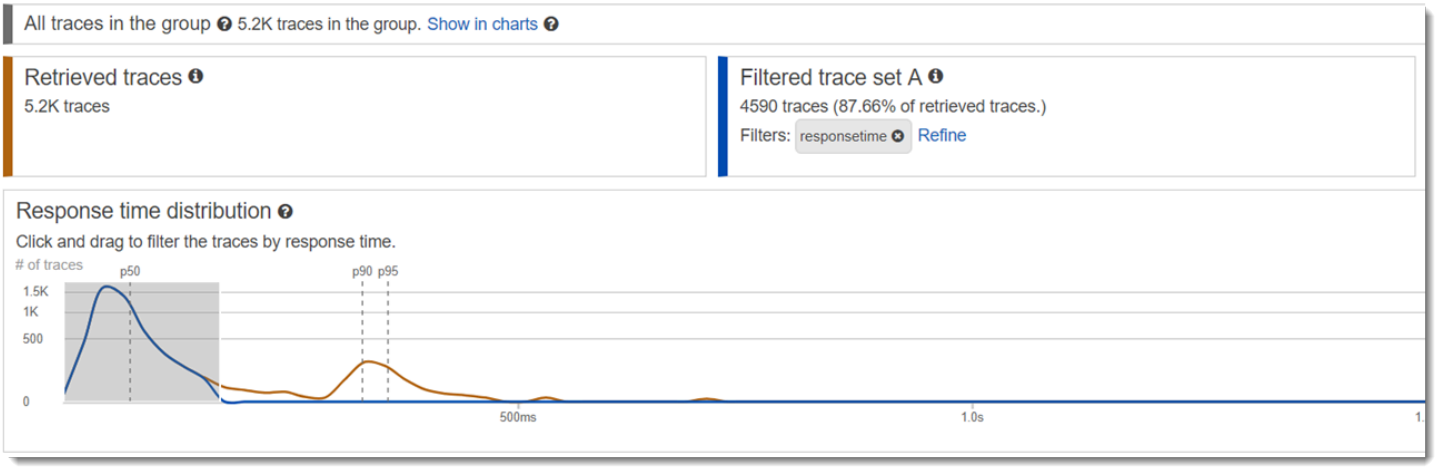
El mapa de rastros indica el estado de cada nodo coloreándolo en función de la proporción entre las llamadas correctas y los errores. Cuando vea un porcentaje en rojo en el nodo, eso indica un error. Utilice la consola de X-Ray Analytics para investigar el problema.

Para obtener más información acerca de cómo leer el mapa de rastros, consulte Consulta del mapa de rastro.

Identificar los picos de tiempo de respuesta

Mediante la distribución del tiempo de respuesta, puede observar los picos de tiempo de respuesta. Al seleccionar el pico en el tiempo de respuesta, las tablas que se encuentran debajo de los gráficos se actualizarán para mostrar las métricas asociadas, como los códigos de estado.

Al hacer clic y arrastrar, X-Ray selecciona y crea un filtro. Se muestra sombreado en gris encima de las líneas gráficas. Ahora puede arrastrar ese elemento resaltado a la izquierda o a la derecha por la distribución para actualizar su selección y filtro.



Consultar todos los registros de seguimiento marcados con un código de estado

Puede analizar los registros de seguimiento dentro del pico seleccionado mediante las tablas de métricas que se encuentran debajo de los gráficos. Al hacer clic en una fila de la tabla HTTP STATUS CODE, se crea automáticamente un filtro en el conjunto de datos de trabajo. Por ejemplo, puede ver todos los registros de seguimiento de código de estado 500. Esto crea una etiqueta de filtro en el icono del conjunto de registros de seguimiento llamado `http.status`.

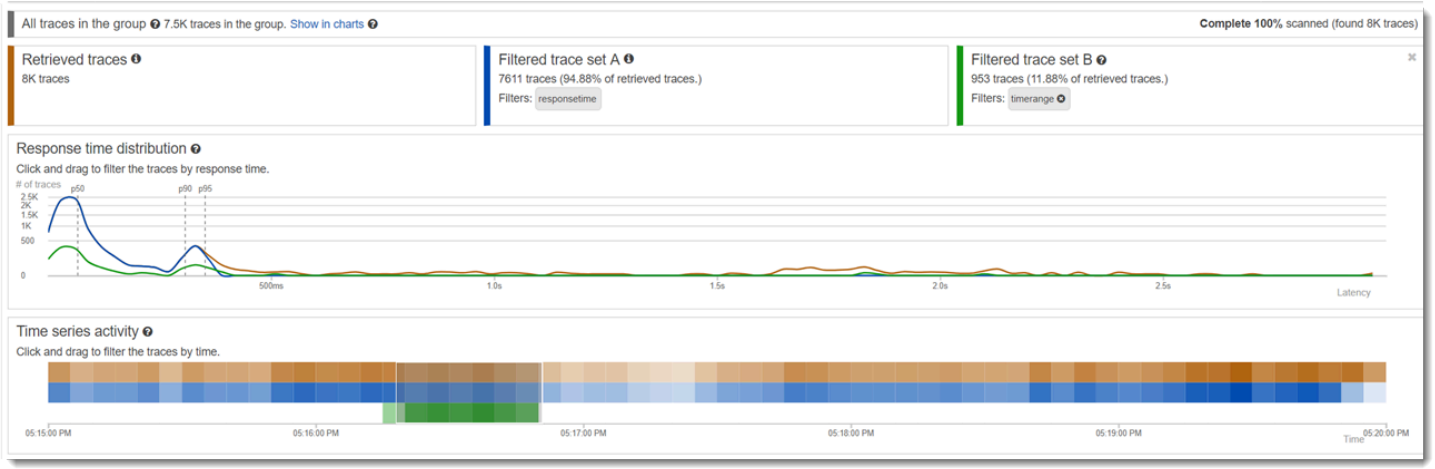
Consultar todos los elementos de un subgrupo y asociados a un usuario

Examine el conjunto de errores en función del usuario, URL, causa raíz del tiempo de respuesta u otros atributos predefinidos. Por ejemplo, para filtrar más el conjunto de registros de seguimiento con un código de estado 500, seleccione una fila de la tabla USERS. Se obtienen dos etiquetas de filtro en el icono del conjunto de registros de seguimiento: `http.status`, tal y como se designó anteriormente, y `user`.

Comparar dos conjuntos de registros de seguimiento con criterios diferentes

Compare varios usuarios y sus solicitudes POST para encontrar otras discrepancias y correlaciones. Aplique su primer conjunto de filtros. Estos filtros se definen por una línea azul en la distribución del tiempo de respuesta. A continuación, seleccione Compare (Comparar). Inicialmente, esto crea una copia de los filtros del conjunto de registros de seguimiento A.

Para continuar, defina un nuevo conjunto de filtros que se aplique al conjunto de registros de seguimiento B. Este segundo conjunto se representa mediante una línea verde. En el siguiente ejemplo se muestran diferentes líneas en función de la combinación de colores azul y verde.



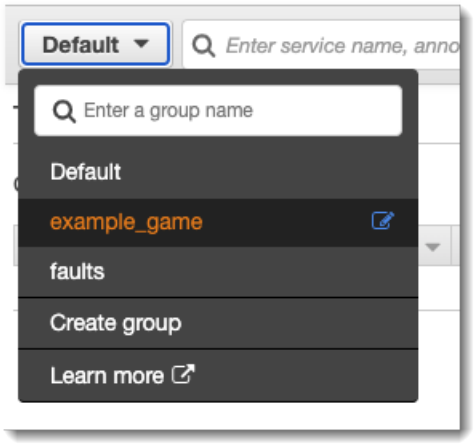
Identificar un registro de seguimiento de su interés y ver sus detalles

Cuando reduzca el alcance mediante los filtros de la consola, la lista de registros de seguimiento de debajo de las tablas de métricas le resultará más útil. La tabla de lista de registros de seguimiento combina información de URL, USER (USUARIO) y STATUS CODE (CÓDIGO DE ESTADO) en una sola consulta. Para obtener más información, seleccione una fila de esta tabla para abrir la página de detalles del registro de seguimiento y ver su escala de tiempo y sus datos sin procesar.

Configuración de grupos

Los grupos son una colección de rastros que se definen mediante una expresión de filtro. Puede utilizar grupos para generar gráficos de servicios adicionales y suministrar métricas de Amazon CloudWatch. Puede usar la consola de AWS X-Ray o la API de X-Ray con el fin de crear y administrar grupos para sus servicios. En este tema se describe cómo crear y administrar grupos con la consola de X-Ray. Para obtener información sobre cómo administrar grupos con la API de X-Ray, consulte Grupos.

Puede crear grupos de rastros para rastros, análisis o mapas de rastros. Al crear un grupo, este pasa a estar disponible como filtro en el menú desplegable del grupo en las tres páginas: Mapa de rastros, Rastros y Análisis.



Los grupos se identifican por su nombre o un nombre de recurso de Amazon (ARN) y contienen una expresión de filtro. El servicio compara los registros de seguimiento de entrada con la expresión y los almacena en consecuencia. Para obtener más información acerca de cómo crear una expresión de filtro, consulte Uso de expresiones de filtro.

La actualización de la expresión de filtro de un grupo no cambia los datos que ya se han registrado. La actualización se aplica únicamente a los rastros posteriores. Esto puede dar lugar a un gráfico que combina la expresión nueva con la anterior. Para evitarlo, elimine el grupo actual y cree uno nuevo.

> **Note**
>
> Los grupos se facturan por el número de rastros recuperados que coinciden con la expresión de filtro. Para más información, consulte Precios de AWS X-Ray.

Temas

- Creación de un grupo
- Aplicar un grupo
- Editar un grupo
- Clonación de un grupo
- Eliminación de un grupo
- Visualización de las métricas de los grupos en Amazon CloudWatch

Creación de un grupo

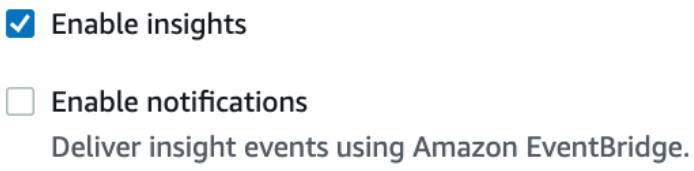
> **Note**
>
> Ahora puede configurar los grupos de X-Ray desde la consola de Amazon CloudWatch. También puede seguir utilizando la consola de X-Ray.

CloudWatch console

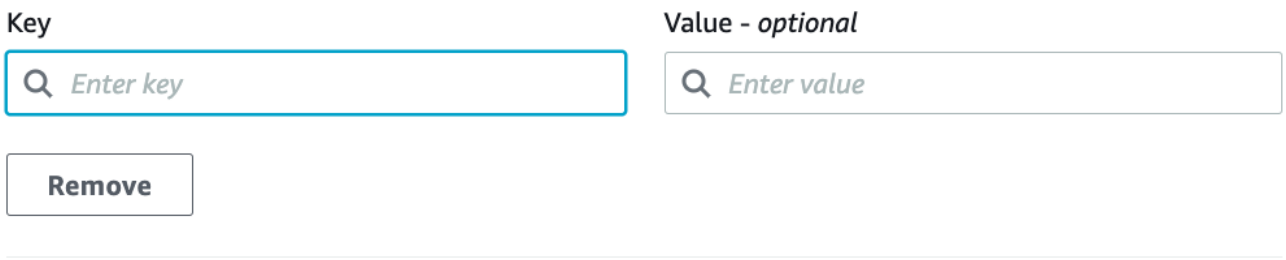
1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Configuración en el panel de navegación izquierdo.
3. Elija Ver ajustes en Grupos dentro de la sección de Rastros de X-Ray.
4. Seleccione Crear grupo en la parte superior de la lista de grupos.
5. En la página Crear grupo escriba un nombre para el grupo. Los nombres de grupo pueden tener un máximo de 32 caracteres y solo pueden contener caracteres alfanuméricos y guiones. En los nombres de grupo se distingue entre mayúsculas y minúsculas.
6. Introduzca una expresión de filtro. Para obtener más información acerca de cómo crear una expresión de filtro, consulte Uso de expresiones de filtro. En el siguiente ejemplo, el grupo filtra los rastros de errores del servicio `api.example.com` y las solicitudes al servicio en las que el tiempo de respuesta fue superior o igual a cinco segundos.

   ```
   fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
   ```

7. En Información, habilite o deshabilite el acceso a la información para el grupo. Para obtener más detalles acerca de la información, consulte Uso de información en X-Ray.

8. En Etiquetas, elija Añadir nueva etiqueta para introducir una clave de etiqueta y, si lo desea, un valor de etiqueta. Siga añadiendo las etiquetas adicionales que desee. Cada clave de etiqueta debe ser única. Para eliminar una etiqueta, elija Eliminar debajo de cada etiqueta. Para obtener más información acerca de las etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.



9. Elija Crear grupo.

X-Ray console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. Abra la página Crear grupo desde la página Grupos del panel de navegación izquierdo o desde el menú de grupos de una de las siguientes páginas: Mapa de rastros, Rastros y Análisis.
3. En la página Crear grupo escriba un nombre para el grupo. Los nombres de grupo pueden tener un máximo de 32 caracteres y solo pueden contener caracteres alfanuméricos y guiones. En los nombres de grupo se distingue entre mayúsculas y minúsculas.
4. Introduzca una expresión de filtro. Para obtener más información acerca de cómo crear una expresión de filtro, consulte Uso de expresiones de filtro. En el siguiente ejemplo, el grupo filtra los rastros de errores del servicio `api.example.com` y las solicitudes al servicio en las que el tiempo de respuesta fue superior o igual a cinco segundos.

```
fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
```

5. En Información, habilite o deshabilite el acceso a la información para el grupo. Para obtener más detalles acerca de la información, consulte Uso de información en X-Ray.

6. En Etiquetas, introduzca una clave de etiqueta y, si lo desea, un valor de etiqueta. Al añadir una etiqueta, aparece una nueva línea para que introduzca otra etiqueta. Cada clave de etiqueta debe ser única. Para eliminar una etiqueta, elija X al final de la fila de la etiqueta. Para obtener más información acerca de las etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.

7. Elija Crear grupo.

Aplicar un grupo

CloudWatch console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Abra una de las siguientes páginas del panel de navegación, en Rastros de X-Ray:
   - Mapa de rastros
   - Rastros
3. Introduzca un nombre de grupo en el Filtrar por grupo de X-Ray. Los datos que se muestran en la página cambian para coincidir con la expresión de filtro establecida en el grupo.

X-Ray console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. Abra una de las siguientes páginas desde el panel de navegación:
   - Mapa de rastros
   - Rastros
   - Analytics
3. En el menú de grupos, elija el grupo que creó en la the section called “Creación de un grupo”. Los datos que se muestran en la página cambian para coincidir con la expresión de filtro establecida en el grupo.

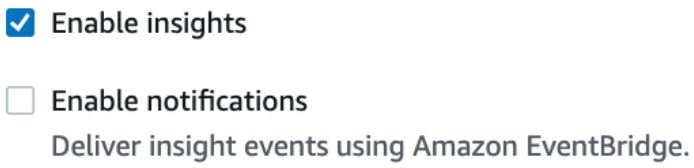
Editar un grupo

CloudWatch console

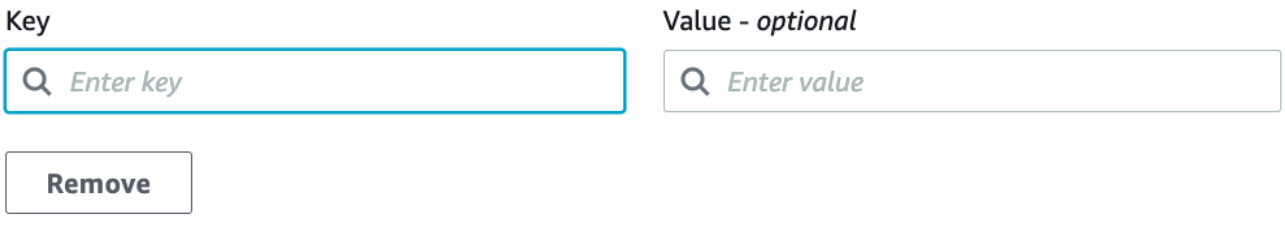
1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Configuración en el panel de navegación izquierdo.
3. Elija Ver ajustes en Grupos dentro de la sección de Rastros de X-Ray.
4. Elija un grupo de la sección Grupos y, a continuación, elija Editar.
5. Aunque no puede cambiar el nombre de un grupo, puede actualizar la expresión de filtro. Para obtener más información acerca de cómo crear una expresión de filtro, consulte Uso de expresiones de filtro. En el siguiente ejemplo, el grupo filtra los rastros de errores del servicio `api.example.com` y las solicitudes cuya dirección URL contenga `example/game` y cuyo tiempo de respuesta fue superior o igual a cinco segundos.

   ```
   fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
   ```

6. En Información, habilite o deshabilite el acceso a la información para el grupo. Para obtener más detalles acerca de la información, consulte Uso de información en X-Ray.

7. En Etiquetas, elija Añadir nueva etiqueta para introducir una clave de etiqueta y, si lo desea, un valor de etiqueta. Siga añadiendo las etiquetas adicionales que desee. Cada clave de etiqueta debe ser única. Para eliminar una etiqueta, elija Eliminar debajo de cada etiqueta. Para obtener más información acerca de las etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.

8. Cuando haya terminado de actualizar el grupo, elija Actualizar grupo.

X-Ray console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. Siga uno de estos pasos para abrir la página Editar grupo.

   a. En la página Grupos, elija el nombre de un grupo para editarlo.

   b. En el menú de grupos de una de las páginas siguientes, elija un grupo y, a continuación, seleccione Editar.

   - Mapa de rastros
   - Rastros
   - Analytics

3. Aunque no puede cambiar el nombre de un grupo, puede actualizar la expresión de filtro. Para obtener más información acerca de cómo crear una expresión de filtro, consulte Uso de expresiones de filtro. En el siguiente ejemplo, el grupo filtra los rastros de errores del servicio `api.example.com` y las solicitudes cuya dirección URL contenga `example/game` y cuyo tiempo de respuesta fue superior o igual a cinco segundos.

   ```
   fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
   ```

4. En Conocimientos, habilite o deshabilite los conocimientos y las notificaciones de conocimientos para el grupo. Para obtener más detalles acerca de la información, consulte Uso de información en X-Ray.

5. En Etiquetas, edite las claves y los valores de las etiquetas. Cada clave de etiqueta debe ser única. Los valores de las etiquetas son opcionales; puede eliminarlos si lo desea. Para eliminar una etiqueta, elija X al final de la fila de la etiqueta. Para obtener más información acerca de las etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.

6. Cuando haya terminado de actualizar el grupo, elija Actualizar grupo.

Clonación de un grupo

Al clonar un grupo, se crea un grupo nuevo que tiene la expresión de filtro y las etiquetas de un grupo existente. Cuando clona un grupo, el grupo nuevo tiene el mismo nombre que el grupo desde el que lo clone, con `-clone` anexado al nombre.

CloudWatch console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Configuración en el panel de navegación izquierdo.
3. Elija Ver ajustes en Grupos dentro de la sección de Rastros de X-Ray.
4. Elija un grupo de la sección Grupos y, a continuación, elija Clonar.
5. En la página Crear grupo, el nombre del grupo es *`group-name`*`-clone`. Si lo desea, puede introducir un nuevo nombre para el grupo. Los nombres de grupo pueden tener un máximo de 32 caracteres y solo pueden contener caracteres alfanuméricos y guiones. En los nombres de grupo se distingue entre mayúsculas y minúsculas.
6. Puede conservar la expresión de filtro del grupo existente o, si lo desea, introducir una nueva expresión de filtro. Para obtener más información acerca de cómo crear una expresión de filtro, consulte Uso de expresiones de filtro. En el siguiente ejemplo, el grupo filtra los rastros de errores del servicio `api.example.com` y las solicitudes al servicio en las que el tiempo de respuesta fue superior o igual a cinco segundos.

   ```
   service("api.example.com") { fault = true OR responsetime >= 5 }
   ```

7. En Etiquetas, edite las claves y los valores de las etiquetas, si es necesario. Cada clave de etiqueta debe ser única. Los valores de las etiquetas son opcionales; puede eliminarlos si lo desea. Para eliminar una etiqueta, elija X al final de la fila de la etiqueta. Para obtener más información acerca de las etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.
8. Elija Crear grupo.

X-Ray console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. Abra la página Grupos desde el panel de navegación izquierdo y elija el nombre del grupo que desee clonar.
3. Seleccione Clonar grupo en el menú Acciones.
4. En la página Crear grupo, el nombre del grupo es *group-name*`-clone`. Si lo desea, puede introducir un nuevo nombre para el grupo. Los nombres de grupo pueden tener un máximo de 32 caracteres y solo pueden contener caracteres alfanuméricos y guiones. En los nombres de grupo se distingue entre mayúsculas y minúsculas.
5. Puede conservar la expresión de filtro del grupo existente o, si lo desea, introducir una nueva expresión de filtro. Para obtener más información acerca de cómo crear una expresión de filtro, consulte Uso de expresiones de filtro. En el siguiente ejemplo, el grupo filtra los rastros de errores del servicio `api.example.com` y las solicitudes al servicio en las que el tiempo de respuesta fue superior o igual a cinco segundos.

   ```
   service("api.example.com") { fault = true OR responsetime >= 5 }
   ```
6. En Etiquetas, edite las claves y los valores de las etiquetas, si es necesario. Cada clave de etiqueta debe ser única. Los valores de las etiquetas son opcionales; puede eliminarlos si lo desea. Para eliminar una etiqueta, elija X al final de la fila de la etiqueta. Para obtener más información acerca de las etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.
7. Elija Crear grupo.

Eliminación de un grupo

Siga los pasos de esta sección para eliminar un grupo. No se puede eliminar el grupo Predeterminado.

CloudWatch console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Configuración en el panel de navegación izquierdo.
3. Elija Ver ajustes en Grupos dentro de la sección de Rastros de X-Ray.

4. Elija un grupo de la sección Grupos y, a continuación, elija Eliminar.
5. Cuando se le pida confirmación, elija Eliminar.

X-Ray console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. Abra la página Grupos desde el panel de navegación izquierdo y elija el nombre del grupo que desee eliminar.
3. En el menú Acciones, elija Eliminar.
4. Cuando se le pida confirmación, elija Eliminar.

Visualización de las métricas de los grupos en Amazon CloudWatch

Una vez que se crea un grupo, los rastros de entrada se comparan con la expresión de filtro del grupo a medida que se almacenan en el servicio de X-Ray. Las métricas para el número de rastros que coinciden con cada criterio se publican en Amazon CloudWatch cada minuto. Al seleccionar Ver métrica en la página Editar grupo, se abre la consola de CloudWatch en la página Métrica. Para obtener más información sobre cómo utilizar las métricas de CloudWatch, consulte Uso de métricas de Amazon CloudWatch en la Guía del usuario de Amazon CloudWatch.

CloudWatch console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Configuración en el panel de navegación izquierdo.
3. Elija Ver ajustes en Grupos dentro de la sección de Rastros de X-Ray.
4. Elija un grupo de la sección Grupos y, a continuación, elija Editar.
5. En la página Editar grupo, elija Ver métrica.

   La página Métricas de la consola de CloudWatch se abre en una nueva pestaña.

X-Ray console

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.

2. Abra la página Grupos desde el panel de navegación izquierdo y elija el nombre del grupo cuyas métricas desee ver.
3. En la página Editar grupo, elija Ver métrica.

   La página Métricas de la consola de CloudWatch se abre en una nueva pestaña.

Configuración de reglas de muestreo de

Puede usar la AWS X-Ray consola para configurar las reglas de muestreo para sus servicios. La AWS distribución OpenTelemetry, el SDK de X-Ray y los Servicios de AWS que admiten el rastreo activo con configuraciones de muestreo utilizan reglas de muestreo para determinar qué solicitudes registrar.

Temas

Configuración de reglas de muestreo de

Puede configurar el muestreo para los siguientes casos de uso:

- Punto de entrada de API Gateway: API Gateway admite el muestreo y el rastreo activo. Para habilitar el rastreo activo en una etapa de la API, consulte Compatibilidad de Amazon API Gateway con el rastreo activo para AWS X-Ray.
- AWS AppSync— AWS AppSync admite el muestreo y el rastreo activo. Para habilitar el rastreo activo en las AWS AppSync solicitudes, consulte Rastreo con X-Ray AWS.
- AWS Step Functions— AWS Step Functions admite el muestreo y el rastreo activo. Para habilitar el rastreo activo en las máquinas AWS Step Functions estatales, consulte Rastreo de rayos X en Step Functions.

- AWS Distribución de instrumentos para OpenTelemetry plataformas informáticas: cuando se utilizan plataformas informáticas como Amazon EC2, Amazon ECS o AWS Elastic Beanstalk, se admite el muestreo cuando la aplicación se ha equipado con la última versión de AWS Distro for o el SDK de OpenTelemetry X-Ray.

Personalización de reglas de muestreo

Al personalizar las reglas de muestreo, puede controlar la cantidad de datos que va a registrar. También puede modificar el comportamiento del muestreo sin modificar ni volver a implementar el código. Las reglas de muestreo indican a la AWS Distro for OpenTelemetry (ADOT) o al X-Ray SDK cuántas solicitudes deben grabar para un conjunto de criterios. De forma predeterminada, el SDK registra la primera solicitud cada segundo y el cinco por ciento de las solicitudes adicionales. Una petición por segundo es el depósito. Esto garantiza que se registre al menos un registro de seguimiento cada segundo mientras el servicio atienda solicitudes. El cinco por ciento es el porcentaje al que se muestrean las solicitudes adicionales más allá del tamaño del depósito.

Puede configurar el SDK de X-Ray para leer reglas de muestreo desde un documento JSON que incluya con su código. Sin embargo, cuando ejecute varias instancias de su servicio, cada instancia realiza el muestreo de manera independiente. Esto hace que el porcentaje total de solicitudes muestreadas aumente, ya que los depósitos de todas las instancias se suman de forma efectiva. Además, para actualizar las reglas de muestreo locales, tiene que volver a implementar su código.

Al definir las reglas de muestreo en la consola de X-Ray y configurar el SDK para leer reglas desde el servicio de X-Ray, puede evitar ambos problemas. El servicio administra el depósito de cada regla y asigna cuotas a cada instancia de su servicio para distribuir el depósito de manera uniforme, en función del número de instancias que se ejecuten. El límite del depósito se calcula de acuerdo con las reglas que haya establecido. Dado que las reglas están configuradas en el servicio, puede administrar las reglas sin realizar implementaciones adicionales.

> **Note**
>
> Al configurar las reglas de muestreo, es fundamental entender que el muestreo por rayos X está «basado en los padres». Esto significa que la decisión de muestreo se toma una sola vez, por lo general, el primer X-Ray-enabled servicio que gestiona la solicitud (el servicio «raíz»).
> Si un servicio descendente recibe una solicitud que ya contiene una decisión de muestreo de una empresa matriz anterior, respetará esa decisión independientemente de sus propias reglas de muestreo coincidentes.

- Cuándo se aplican las normas: sus reglas de muestreo personalizadas solo entran en vigor en los servicios en los que aún no se ha tomado una decisión de muestreo. Por lo general, esto se aplica a:
  - El punto de entrada de su aplicación (p. ej., una API Gateway, un Load Balancer o el primer microservicio instrumentado).
  - Procesos o trabajadores asíncronos que inician una nueva trayectoria.
- Error común: si crea una regla de muestreo estricta para el «Servicio B», pero siempre se llama «Servicio B» por «Servicio A», es probable que su regla para el Servicio B nunca se aplique porque simplemente se trata de seguir la decisión adoptada por el Servicio A. Para cambiar el muestreo de trazas para este flujo de trabajo, debe configurar la regla de muestreo para el servicio raíz (Servicio A).

> **Note**
>
> X-Ray aplica las reglas de muestreo en la medida de lo posible y, en algunos casos, el porcentaje de muestreo efectivo puede no coincidir exactamente con las reglas de muestreo configuradas. Sin embargo, con el tiempo, el número de solicitudes muestreadas debería estar cerca del porcentaje configurado.

Ahora puede configurar las reglas de muestreo de X-Ray desde la CloudWatch consola de Amazon.

Para configurar las reglas de muestreo en la CloudWatch consola

1. Inicie sesión en Consola de administración de AWS y abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. Selecciona Configuración en el panel de navegación izquierdo, debajo de Configuración.
3. Seleccione Ver configuración en Reglas de muestreo en la sección de Rastros de X-Ray.
4. Para crear una regla, elija Crear regla de muestreo.

   Para editar una regla, elija la regla y, a continuación, elija Editar.

   Para eliminar una regla, elija la regla y, a continuación, elija la opción Eliminar.

Opciones de reglas de muestreo

Las siguientes opciones están disponibles para cada regla. En los valores de cadena se pueden usar caracteres comodín para buscar coincidencias de un solo carácter (?) o cero o más caracteres (*).

Opciones de reglas de muestreo

- Nombre de la regla (cadena): un nombre único para la regla.
- Prioridad (entero comprendido entre el 1 y 9999): prioridad de la regla de muestreo. Los servicios evalúan las reglas en orden ascendente de prioridad y toman una decisión de muestreo con la primera regla coincidente.
- Depósito (entero no negativo): número fijo de solicitudes coincidentes que se van a instrumentar por segundo, antes de aplicar el porcentaje fijo. Los servicios no utilizan directamente el depósito, sino que se aplica a todos los servicios que usan la regla en su conjunto.
- Porcentaje (entero comprendido entre el 0 y el 100): porcentaje de solicitudes coincidentes que se van instrumentar, una vez que se ha agotado el depósito. Al configurar una regla de muestreo en la consola, elija un porcentaje entre 0 y 100. Al configurar una regla de muestreo en un SDK de cliente mediante un documento JSON, proporcione un valor porcentual entre 0 y 1.
- Nombre del servicio (cadena): el nombre del servicio instrumentado, tal como aparece en el mapa de rastros.
  - SDK de X-Ray: el nombre del servicio que se configura en la grabadora.
  - Amazon API Gateway: *`api-name`*/*`stage`*.
- Tipo de servicio (cadena): el tipo de servicio, tal como aparece en el mapa de rastros. Para el SDK de X-Ray, defina el tipo de servicio aplicando el complemento adecuado:
  - `AWS::ElasticBeanstalk::Environment`— Un AWS Elastic Beanstalk entorno (complemento).
  - `AWS::EC2::Instance`— Una EC2 instancia de Amazon (plugin).
  - `AWS::ECS::Container`: un contenedor de Amazon ECS (complemento).
  - `AWS::EKS::Container`— Un contenedor Amazon EKS (plugin).
  - `AWS::APIGateway::Stage`: una etapa de Amazon API Gateway.
  - `AWS::AppSync::GraphQLAPI` — Una solicitud AWS AppSync de API.
  - `AWS::StepFunctions::StateMachine`— Una máquina de AWS Step Functions estados.
- Host (cadena): nombre de host del encabezado de host HTTP.
- Método HTTP (cadena): el método de la solicitud HTTP.

- Ruta URL (cadena): la ruta URL de la solicitud.
  - SDK de X-Ray: la parte de la ruta de la URL de la solicitud HTTP.
- ARN del recurso (cadena): el ARN del AWS recurso que ejecuta el servicio.
  - SDK de X-Ray: no compatible. El SDK solo puede utilizar reglas con Resource ARN (ARN de recurso) configurado en `*`.
  - Amazon API Gateway: el ARN de etapa.
- (Opcional) Atributos (clave y valor): atributos de segmento que se conocen cuando se toma la decisión de muestreo.
  - SDK de X-Ray: no compatible. El SDK omite las reglas que especifican atributos.
  - Amazon API Gateway: encabezados de la solicitud HTTP original.
- (Opcional) SamplingRateBoost(objeto): controla el comportamiento de mejora del muestreo provocado por anomalías.
  - MaxRate (número entero entre 0 y 100): la frecuencia máxima de muestreo (0,0—1,0) Los rayos X pueden aumentar durante las anomalías
  - CooldownWindowMinutes (entero mayor que 0): intervalo de tiempo (minutos) en el que solo se puede activar un impulso, lo que evita que los amplificadores sean continuos

Ejemplos de reglas de muestreo

Example– Regla predeterminada sin depósito y con un porcentaje bajo

Puede modificar el depósito predeterminado de la regla y el porcentaje. La regla predeterminada se aplica a las solicitudes que no coinciden con cualquier otra regla.

- Depósito: **`0`**
- Porcentaje: **`5`** (**`0.05`** si se configura mediante un documento JSON)

Example– Regla de depuración para rastrear todas las solicitudes para una ruta problemática

Una regla de alta prioridad aplicada temporalmente para depuración.

- Nombre de la regla: **`DEBUG - history updates`**
- Prioridad: **`1`**
- Depósito: **`1`**
- Porcentaje: **`100`** (**`1`** si se configura mediante un documento JSON)

- Nombre del servicio: **`Scorekeep`**
- Tipo de servicio: **`*`**
- Host: **`*`**
- Método HTTP: **`PUT`**
- Ruta URL: **`/history/*`**
- ARN de recurso: **`*`**

Example— Tasa mínima más alta para POSTs

- Nombre de la regla: **`POST minimum`**
- Prioridad: **`100`**
- Depósito: **`10`**
- Porcentaje: **`10`** (**`.1`** si se configura mediante un documento JSON)
- Nombre del servicio: **`*`**
- Tipo de servicio: **`*`**
- Host: **`*`**
- Método HTTP: **`POST`**
- Ruta URL: **`*`**
- ARN de recurso: **`*`**

Example Habilitación de un aumento provocado por anomalías

Configure una regla que desencadene un aumento de muestreo de hasta un 50 % durante las anomalías, con un tiempo de reutilización de 10 minutos.

- Nombre de la regla: **`MyAdaptiveRule`**
- Prioridad: **`100`**
- Depósito: **`1`**
- FixedRate: **`0.0510`**
- Nombre del servicio: **`*`**
- Tipo de servicio: **`*`**
- Host: **`*`**

- Método HTTP: **`POST`**
- Ruta URL: **`*`**
- maxRate: **`0.5`**
- cooldownWindowMinutes: **`10`**

Configuración del servicio para utilizar reglas de muestreo

La AWS distribución para OpenTelemetry (ADOT) y el SDK de X-Ray requieren una configuración adicional para utilizar las reglas de muestreo que se configuran en la consola. Consulte el tema de configuración para su lenguaje para obtener más información sobre cómo configurar una estrategia de muestreo.

- Java: Uso del muestreo remoto de rayos X con ADOT Java
- Go: Configurar el muestreo con ADOT Go
- Node.js: Uso del muestreo remoto de rayos X con ADOT JavaScript
- Python: Uso del muestreo remoto de rayos X con ADOT Python
- Ruby: Reglas de muestreo
- .NET: Uso del muestreo remoto de rayos X con ADOT.NET

Para API Gateway, consulte: Compatibilidad de Amazon API Gateway con el rastreo activo para AWS X-Ray.

Visualización de resultados de muestreo

La página de Muestreo de la consola de X-Ray muestra información detallada sobre cómo los servicios del usuario utilizan cada regla de muestreo.

La columna Trend (Tendencia) muestra cómo se ha utilizado la regla en los últimos minutos. Cada columna muestra las estadísticas para una ventana de 10 segundos.

Estadísticas de muestreo

- Regla coincidente total: el número de solicitudes que coincidieron con esta regla. Este número no incluye solicitudes que podrían haber coincidido con esta regla, pero coincidieron primero con una regla de prioridad más alta.
- Total muestreado: el número de solicitudes registradas.

- Muestreadas con porcentaje fijo: número de solicitudes muestreadas aplicando el porcentaje fijo de la regla.
- Muestreado con límite de depósito: el número de solicitudes muestreadas utilizando una cuota asignada por X-Ray.
- Préstamos del depósito: número de solicitudes muestreadas tomando prestado del depósito. La primera vez que un servicio hace coincidir una solicitud con una regla, X-Ray aún no le ha asignado una cuota. Sin embargo, si el depósito es al menos 1, el servicio toma prestado un rastro por segundo hasta que X-Ray asigna una cuota.

Para obtener más información acerca de cómo utilizar las estadísticas de muestreo y cómo utilizan las reglas de muestreo los servicios, consulte Using sampling rules with the X-Ray API (Uso de reglas de muestreo con la API de X-Ray).

Siguientes pasos

Puede usar la API de X-Ray para administrar reglas de muestreo. Con la API, puede crear y actualizar las reglas mediante programación de forma programada o en respuesta a alarmas o notificaciones. Consulte Configuración de las opciones de muestreo, grupos y cifrado con la API de AWS X-Ray para obtener instrucciones y ejemplos de reglas adicionales.

The AWS Distro for OpenTelemetry, X-Ray SDK y Servicios de AWS también utilizan la API de X-Ray para leer las reglas de muestreo, informar los resultados del muestreo y obtener los objetivos de muestreo. Los servicios deben realizar un seguimiento de la frecuencia con la que se aplica cada regla, evaluar las reglas en función de la prioridad y tomar prestado del depósito cuando una solicitud coincide con una regla para la que X-Ray no ha asignado aún una cuota al servicio. Para obtener más información acerca de cómo utiliza un servicio la API para muestreo, consulte Using sampling rules with the X-Ray API (Uso de reglas de muestreo con la API de X-Ray).

Cuando la AWS Distro for OpenTelemetry o el SDK de X-Ray llaman al muestreo APIs, utilizan el CloudWatch agente como proxy. Si ya utiliza el puerto TCP 2000, puede configurar el agente para que ejecute el proxy en un puerto diferente. Consulte la guía de instalación del CloudWatch agente para obtener más información.

Configuración de muestreo flexible

La falta de rastros críticos durante los picos de anomalías puede dificultar el análisis de la causa raíz. Sin embargo, mantener altas tasas de muestreo resulta caro. El muestreo adaptativo de X-Ray proporciona una visibilidad completa de las anomalías y controla los costos durante las operaciones

normales. Con el muestreo adaptativo, se establece una frecuencia de muestreo máxima y X-Ray se ajusta automáticamente dentro de ese límite. X-Ray calcula el impulso mínimo necesario para capturar los rastros de errores. Si la velocidad de referencia captura suficientes datos, no se produce ningún aumento. Solo pagará por el muestreo adicional cuando sea necesario.

Beneficios de utilizar el muestreo adaptativo:

- Visibilidad completa de los incidentes: obtenga un seguimiento completo de los incidentes sin intervención manual. X-Ray ajusta automáticamente las velocidades de muestreo para capturar los rastros de error y, a continuación, vuelve a las tasas normales.
- Visibilidad de la causa raíz: compruebe siempre el origen de los problemas. X-Ray captura datos de errores críticos incluso cuando no se desencadena el muestreo de rastros completos.
- Optimice los costos: los breves aumentos de muestreo (hasta 1 minuto) y los periodos de recuperación automáticos evitan el sobremuestreo. Solo pagará por los datos que necesite para diagnosticar problemas.

Temas

Plataformas y SDK compatibles

SDK compatible: el muestreo adaptativo requiere la versión más reciente del ADOT SDK.

Lenguaje compatible: Java (versión 2.11.5 o superior)

La aplicación debe estar equipada con el ADOT SDK compatible y ejecutarse junto con el agente de Amazon CloudWatch o el recopilador de OpenTelemetry.

Por ejemplo, Amazon EC2, Amazon ECS y Amazon EKS son plataformas comunes en las que las señales de aplicación de AWS proporcionan orientación para habilitar ADOT SDK y el agente de Amazon CloudWatch.

Elección del enfoque de muestreo adaptativo

El muestreo adaptativo admite dos enfoques: potenciación del muestreo y captura de intervalos de anomalías. Se pueden aplicar de forma independiente o se pueden combinar entre sí.

Potenciación del muestreo

La potenciación del muestreo adaptativo se basa en las reglas de muestreo y funciona con el modelo de muestreo existente basado en encabezados de X-Ray. El muestreo basado en encabezados significa que las decisiones de muestreo se toman en el servicio raíz y el indicador de muestreo se transmite posteriormente a todos los servicios de la cadena de llamadas.

- Aumento basado en reglas: el impulso siempre está vinculado a una regla específica de muestreo de X-Ray. Cada regla puede definir su propia velocidad máxima de aumento y comportamiento de enfriamiento.
- Muestreo basado en encabezados: significa que las decisiones de muestreo se toman en el servicio raíz y el indicador de muestreo se transmite posteriormente a todos los servicios de la cadena de llamadas.
- Impulsado por anomalías: X-Ray se basa en el SDK para informar de las estadísticas de anomalías. Cuando X-Ray detecta anomalías, como errores o una latencia alta, utiliza estas estadísticas para calcular una tasa de aumento adecuada (hasta el máximo configurado).

Informes de anomalías

Todos los servicios de aplicaciones de la cadena de llamadas pueden emitir estadísticas de anomalías a través del SDK necesario:

- Servicio raíz: se debe ejecutar en una plataforma y un SDK compatibles para poder aumentar el muestreo. Si el servicio raíz no es compatible, no se producirá ningún impulso.
- Servicios posteriores: los servicios posteriores solo informan de anomalías; no pueden tomar decisiones de muestreo. Cuando un servicio posterior ejecuta un SDK compatible, las anomalías detectadas pueden desencadenar un aumento del muestreo. Cuando un servicio posterior no es compatible (por ejemplo, si se ejecuta un SDK anterior), las anomalías de ese servicio no desencadenarán un aumento. Estos servicios aún pueden propagar el contexto posterior si siguen la propagación de contexto estándar (como el contexto y equipaje de rastros de W3C). Esto garantiza que los SDK compatibles en otros servicios posteriores puedan detectar anomalías que desencadenen un aumento.

Aumento del tiempo y el alcance

- Retraso de desencadenamiento: puede esperar que el aumento de muestreo comience tan solo 10 segundos después de que X-Ray detecte una anomalía.

- Periodo de refuerzo: después de que X-Ray desencadene un impulso, dura hasta 1 minuto antes de volver a la frecuencia de muestreo base.
- Aumentar el tiempo de enfriamiento: después de que se produzca un aumento, X-Ray no desencadenará otro aumento con la misma regla hasta que haya pasado el periodo de enfriamiento.

  Por ejemplo, si establece `cooldown` en 10 minutos, una vez que finalice un aumento, no se podrá desencadenar ningún nuevo impulso hasta que pasen los siguientes 10 minutos.

  Caso especial: si establece `cooldown` en 1 minuto, y dado que un impulso en sí mismo puede durar hasta 1 minuto, los aumentos pueden desencadenarse de forma continua si la anomalía persiste.

> **Note**
>
> Use los SDK y las plataformas compatibles para el servicio raíz. La potenciación del muestreo solo funciona con los SDK y las plataformas compatibles. Aunque la potenciación del muestreo tiene una alta probabilidad de captar rastros de anomalías, es posible que no capte todos los rastros de anomalías.

Aumento de la visibilidad

Cuando se configura una regla de muestreo con un aumento de muestreo adaptativo, X-Ray emite automáticamente métricas vendidas que le permiten supervisar la actividad del aumento.

- Nombre de métrica – `SamplingRate`
- Dimensión: `RuleName` (establecida con el nombre real de la regla)

Cada regla con `SamplingRateBoost` habilitado publicará su frecuencia de muestreo efectiva, incluida la frecuencia de referencia y cualquier aumento temporal. Esto le permite:

- Realización de un seguimiento de cuándo se desencadenan los aumentos
- Supervisión de la frecuencia de muestreo efectiva de cada regla
- Correlación de los aumentos con las anomalías de las aplicaciones (como picos de error o eventos de latencia)

Puede ver estas métricas en Métricas de Amazon CloudWatch, en el espacio de nombres AWS/X-Ray. El valor de la métrica es un número de punto flotante entre 0 y 1, que representa la frecuencia de muestreo efectiva.

Configuración de potenciación de muestreo mediante reglas de muestreo de X-Ray

Puede habilitar el muestreo adaptativo directamente en las reglas de muestreo de X-Ray existentes agregando un campo de `SamplingRateBoost` nuevo. Para obtener más información, consulte Configuración de reglas de muestreo. Esto proporciona una forma centralizada de habilitar el muestreo adaptativo sin modificar el código de la aplicación ni aplicar la implementación de la aplicación. Al habilitar el muestreo adaptativo, X-Ray aumenta automáticamente el muestreo durante anomalías como picos de error o valores atípicos de latencia, a la vez que mantiene las frecuencias de muestreo dentro del máximo configurado. `SamplingRateBoost` se puede aplicar a cualquier regla de muestreo personalizada, excepto a la regla de muestreo `Default`.

El campo `SamplingRateBoost` define el límite superior y el comportamiento del muestreo provocado por anomalías.

```
"SamplingRateBoost": {
  "MaxRate": 0.25,
  "CooldownWindowMinutes": 10
}
```

`MaxRate` define la frecuencia de muestreo máxima que aplicará X-Ray cuando detecte anomalías. El intervalo de valores es `0.0` a `1.0`. Por ejemplo, `"MaxRate": 0.25` permite que el muestreo aumente hasta un 25 % de las solicitudes durante un periodo de anomalías. X-Ray determina la frecuencia adecuada entre la línea base y la máxima, en función de la actividad de la anomalía.

`CooldownWindowMinutes` define el intervalo de tiempo (en minutos) en el que solo se puede desencadenar un aumento de la frecuencia de muestreo. Una vez que se produce un aumento, no se permiten más aumentos hasta la siguiente ventana. El tipo de valor es entero (minutos).

Regla de ejemplo con muestreo adaptativo

```
{
  "RuleName": "MyAdaptiveRule",
  "Priority": 1,
```

```
  "ReservoirSize": 1,
  "FixedRate": 0.05,
  "ServiceName": "*",
  "ServiceType": "*",
  "Host": "*",
  "HTTPMethod": "*",
  "URLPath": "*",
  "SamplingRateBoost": {
    "MaxRate": 0.25,
    "CooldownWindowMinutes": 10
  }
}
```

En este ejemplo, el muestreo de referencia es del 5 % (`FixedRate: 0.05`). Durante las anomalías, X-Ray puede aumentar el muestreo hasta un 25 % (`MaxRate: 0.25`). Aumente solo una vez cada 10 minutos.

Configuración de estado anómalo

Cuando no se proporciona una configuración de condición de anomalía, el ADOT SDK utiliza los códigos de error HTTP 5xx como condición de anomalía predeterminada para desencadenar el aumento del muestreo.

También puede refinar las condiciones de las anomalías de forma local en el ADOT SDK compatible mediante variables de entorno. Para obtener más información, consulte Configuración de SDK local.

Captura de intervalos de anomalías

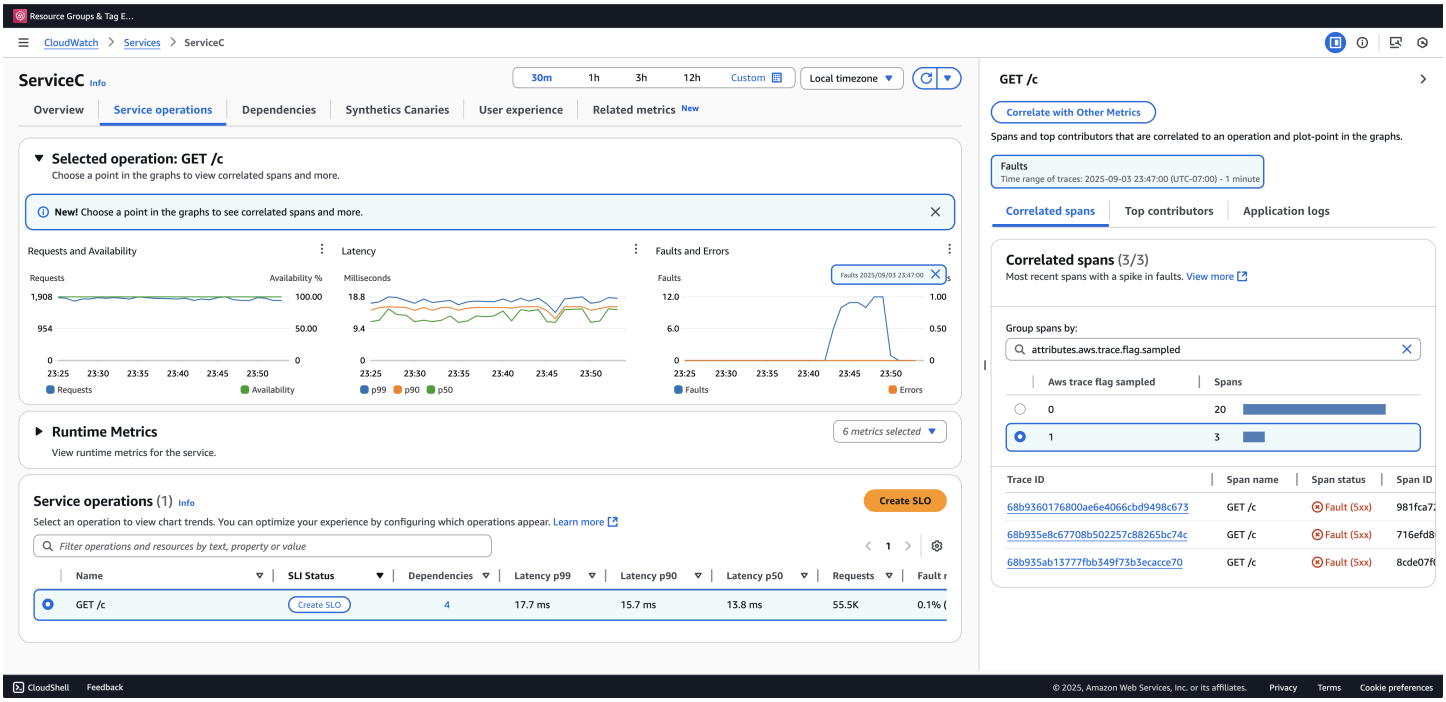
La captura de intervalos de anomalías garantiza que los intervalos críticos que representan anomalías se registren siempre, incluso si no se ha muestreado todo el rastro. Esta característica complementa el impulso del muestreo al centrarse en capturar la anomalía en sí misma, en lugar de aumentar el muestreo para rastros futuros.

Cuando el ADOT SDK detecta una anomalía, emite ese intervalo inmediatamente, independientemente de la decisión de muestreo. Como el SDK solo emite intervalos relacionados con la anomalía, estos rastros son parciales, no transacciones completas integrales.

Una vez que ADOT SDK detecta un intervalo de anomalías, intenta emitir tantos intervalos como sea posible a partir del mismo rastro. Todos los intervalos emitidos en virtud de esta característica se etiquetan con el atributo, `aws.trace.flag.sampled = 0`. Esto le permite distinguir fácilmente los

rastros parciales (captura de anomalías) de los rastros completos (muestreo normal) en la búsqueda y el análisis de las transacciones.

Recomendamos incorporar Búsqueda de transacciones para ver y consultar los rastros parciales. El siguiente ejemplo muestra una página de servicio en la consola de señales de aplicaciones. El ServiceC está configurado con la captura de intervalos de anomalías y forma parte de una cadena de llamadas en la que se aplica el aumento de muestreo. Esta configuración genera rastros completos y parciales. Puede utilizar el atributo `aws.trace.flag.sampled` para distinguir entre los tipos de rastros.



La captura de intervalos de anomalías solo se puede habilitar o personalizar a través de Configuración de SDK local.

Configuración de SDK local

Puede configurar las características de muestreo adaptativo en el ADOT SDK proporcionando una configuración YAML a través de una variable de entorno. La configuración local proporciona un control minucioso sobre las condiciones de anomalía y los umbrales.

Esto es necesario para la captura de intervalos de anomalías y es opcional para personalizar las condiciones de potenciación del muestreo. A continuación, se muestra un ejemplo de la configuración:

```
version: 1.0
anomalyConditions:
  - errorCodeRegex: "^5\\d\\d$"
    usage: both
  - operations:
      - "/api"
    errorCodeRegex: "^429|5\\d\\d$"
    highLatencyMs: 300
    usage: sampling-boost
  - highLatencyMs: 1000
    usage: anomaly-span-capture

anomalyCaptureLimit:
  anomalyTracesPerSecond: 1
```

Las definiciones de los campos son las siguientes:

- `version`: versión esquemática del archivo de configuración
- `anomalyConditions`: define las condiciones en las que se detectan las anomalías y cómo se utilizan
  - `errorCodeRegex`: expresión regular que define qué códigos de estado HTTP se consideran anomalías
  - `operations`: lista de operaciones o puntos de conexión a los que se aplica la condición
  - `highLatencyMs`: umbral de latencia (en milisegundos) por encima del cual los intervalos se consideran anomalías
  - `usage`: define a qué característica se aplica la condición:
    - `both`: se aplica a la potenciación del muestreo y a la captura de intervalos de anomalías (predeterminado si no se especifica el uso)
    - `sampling-boost`: se utiliza solo para desencadenar aumentos de muestreo
    - `anomaly-span-capture`: se utiliza solo para la captura de intervalos de anomalías
- `anomalyCaptureLimit`: define los límites del número de rastros que se emiten con intervalos de anomalías.

  `anomalyTracesPerSecond`: número máximo de rastros con intervalos de anomalías capturados por segundo, para evitar un volumen de intervalo excesivo (el valor predeterminado es 1 si anomalyCaptureLimit no está presente).

Note

- `AnomalyConditions` invalida la condición de anomalía predeterminada para el aumento del muestreo (HTTP 5xx). Si desea retener la condición predeterminada mientras utiliza la configuración local, debe incluirla explícitamente en cualquier elemento de `AnomalyConditions`.
- Para cada elemento `anomalyConditions`:
  - Cuando el campo `operations` se omite, la condición se aplica a todas las operaciones (nivel de servicio)
  - Cuando el campo `operations` está presente pero está configurado en una lista vacía, la condición no se aplica a ninguna operación, por lo que ese elemento no es operativo
  - Si se omite `errorCodeRegex` y `highLatencyMs`, la condición no tiene ningún criterio de anomalía que evaluar, por lo que ese elemento no es operativo
- Relaciones lógicas:
  - Entre los elementos de `anomalyConditions`, la relación es OR.
  - Dentro de un único elemento, varios campos (por ejemplo, `errorCodeRegex` y `highLatencyMs`) se combinan con AND.

    Por ejemplo:

    ```
    errorCodeRegex: "^429|5\\d\\d$"
    highLatencyMs: 300
    ```

    Esta condición significa que el código de estado coincide con 429 o 5xx Y una latencia ≥ 300 ms.

Aplicación de la configuración local al ADOT SDK

Puede aplicar la configuración local al ADOT SDK configurando la variable de entorno `AWS_XRAY_ADAPTIVE_SAMPLING_CONFIG`. El valor debe ser un documento YAML válido (integrado o anidado).

Por ejemplo, Amazon EC2 y Amazon ECS configuran la variable de entorno directamente:

```
AWS_XRAY_ADAPTIVE_SAMPLING_CONFIG="{version: 1.0, anomalyConditions: [{errorCodeRegex:
 \"^500$\", usage: \"sampling-boost\"}, {errorCodeRegex: \"^501$\", usage: \"anomaly-
trace-capture\"}], anomalyCaptureLimit: {anomalyTracesPerSecond: 10}}"
```

Para Amazon EKS, defina la variable de entorno dentro de la especificación del pod como YAML anidado:

```
apiVersion: v1
kind: Pod
metadata:
  name: adot-sample
spec:
  containers:
    - name: adot-app
      image: my-app:latest
      env:
        - name: AWS_XRAY_ADAPTIVE_SAMPLING_CONFIG
          value: |
            version: 1.0
            anomalyConditions:
              - errorCodeRegex: "^500$"
                usage: sampling-boost
              - errorCodeRegex: "^501$"
                usage: anomaly-trace-capture
            anomalyCaptureLimit:
              anomalyTracesPerSecond: 10
```

Enlace profundo de la consola

Puede usar rutas y consultas para establecer un enlace profundo rastros específicos, o bien usar vistas filtradas de rastros y el mapa de rastros.

Páginas de la consola

- Página principal: xray/home#/welcome
- Introducción: xray/home#/getting-started
- Mapa de rastros: xray/home#/service-map

- Rastros: xray/home#/traces

Registros de seguimiento

Puede generar enlaces a las vistas de escala de tiempo, sin procesar y mapa de los rastros individuales.

Escala de tiempo de rastros: `xray/home#/traces/`*`trace-id`*

Datos de rastro sin procesar: `xray/home#/traces/`*`trace-id`*`/raw`

Example– datos de rastro sin procesar

```
https://console.aws.amazon.com/xray/home#/traces/1-57f5498f-d91047849216d0f2ea3b6442/
raw
```

Expresiones de filtro

Enlazan con una lista filtrada de rastros.

Vista de rastros filtrados: `xray/home#/traces?filter=`*`filter-expression`*

Example– expresión de filtro

```
https://console.aws.amazon.com/xray/home#/traces?filter=service("api.amazon.com")
 { fault = true OR responsetime > 2.5 } AND annotation.foo = "bar"
```

Example– expresión de filtro (URL codificada)

```
https://console.aws.amazon.com/xray/home#/traces?filter=service(%22api.amazon.com
%22)%20%7B%20fault%20%3D%20true%20OR%20responsetime%20%3E%202.5%20%7D%20AND
%20annotation.foo%20%3D%20%22bar%22
```

Para obtener más información sobre las expresiones de filtro, consulte Uso de expresiones de filtro.

Intervalo de tiempo

Especifique un intervalo de tiempo o una hora de inicio y una hora de finalización en formato ISO8601. Los rangos de tiempo se indican en UTC y su duración máxima es de seis horas.

Período de tiempo: `xray/home#/`*`page`*`?timeRange=`*`range-in-minutes`*

Example— mapa de rastros de la última hora

```
https://console.aws.amazon.com/xray/home#/service-map?timeRange=PT1H
```

Hora de inicio y finalización: `xray/home#/`*`page`*`?timeRange=`*`start`*`~`*`end`*

Example– intervalo de tiempo con una precisión de segundos

```
https://console.aws.amazon.com/xray/home#/traces?
timeRange=2023-7-01T16:00:00~2023-7-01T22:00:00
```

Example– intervalo de tiempo con una precisión de minutos

```
https://console.aws.amazon.com/xray/home#/traces?
timeRange=2023-7-01T16:00~2023-7-01T22:00
```

Región

Especifique una Región de AWS para enlazar a las páginas que haya en esa región. De lo contrario, la consola le redirige a la última región que ha visitado.

Región: `xray/home`**`?region=`**`region`*`#/page`*

Example– mapa de rastros en el Oeste de EE. UU. (Oregón) (us-west-2)

```
https://console.aws.amazon.com/xray/home?region=us-west-2#/service-map
```

Cuando se incluye una región con otros parámetros de consulta, la consulta de región va antes de la almohadilla y las consultas específicas de X-Ray van después del nombre de página.

Example– mapa de rastros de la última hora en el Oeste de EE. UU. (Oregón) (us-west-2)

```
https://console.aws.amazon.com/xray/home?region=us-west-2#/service-map?timeRange=PT1H
```

En combinación

Example– rastros recientes con un filtro de duración

```
https://console.aws.amazon.com/xray/home#/traces?timeRange=PT15M&filter=duration%20%3E
%3D%205%20AND%20duration%20%3C%3D%208
```

Output

- Página — Rastros
- Intervalo de tiempo: últimos 15 minutos
- Filtro: duración >= 5 Y duración <= 8

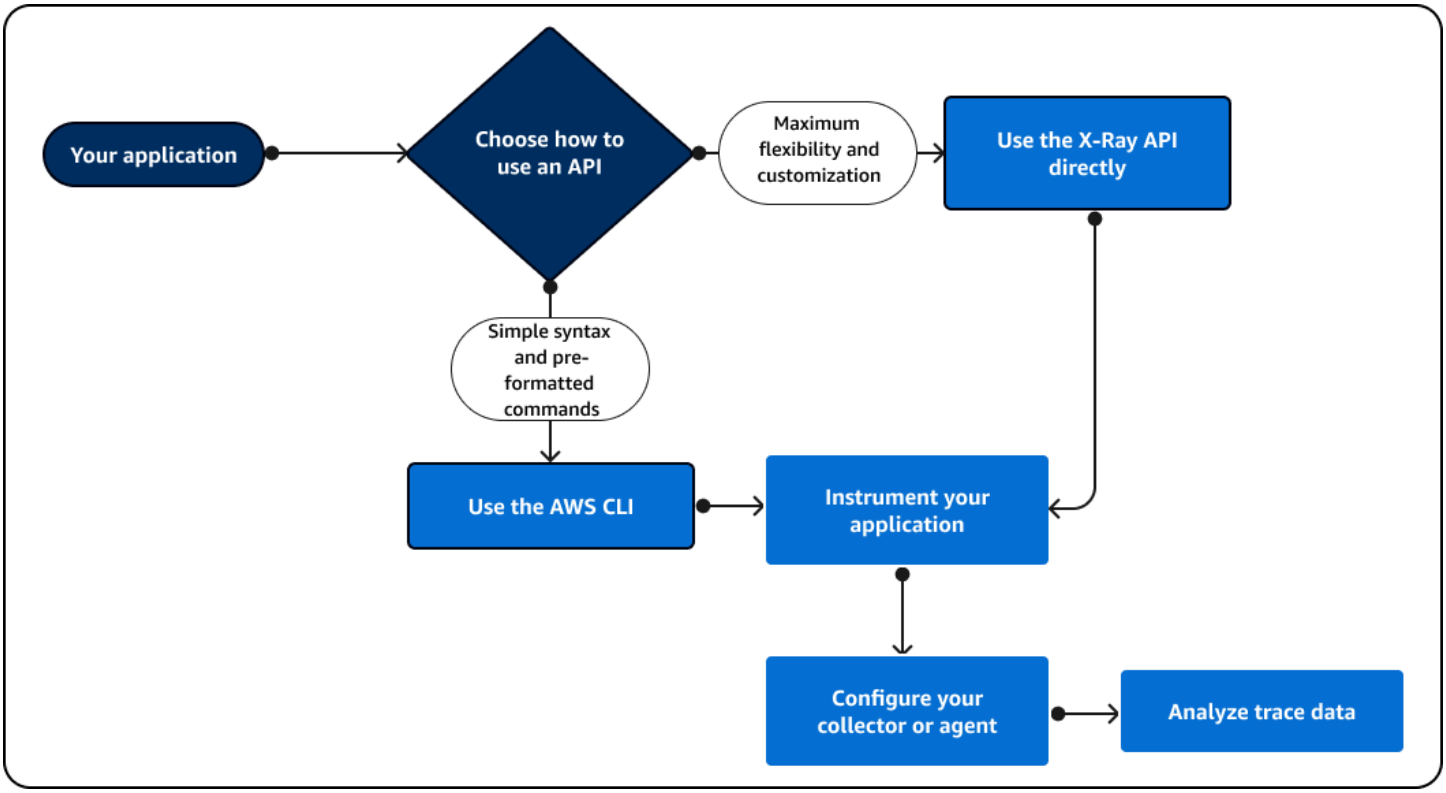
Uso de la API de X-Ray

Si el SDK de X-Ray no es compatible con su lenguaje de programación, puede usar las API de X-Ray directamente o la AWS Command Line Interface (AWS CLI) para llamar a los comandos de la API de X-Ray. Emplee las siguientes directrices para escoger la forma de interactuar con la API:

- Utilice la AWS CLI para simplificar la sintaxis mediante comandos preformateados o con opciones incluidas en su solicitud.
- Utilice la API de X-Ray directamente para flexibilizar y personalizar al máximo las solicitudes que haga a X-Ray.

Si usa la API de X-Ray directamente en lugar de la AWS CLI, deberá parametrizar la solicitud en el formato de datos correcto y es posible que también tenga que configurar la autenticación y la gestión de errores.

El siguiente diagrama muestra una guía para elegir cómo interactuar con la API de X-Ray:



Utilice la API de X-Ray para enviar datos de rastreo directamente a X-Ray. La API de X-Ray admite todas las funciones disponibles en el SDK de X-Ray, incluidas las siguientes acciones comunes:

- PutTraceSegments: carga documentos de segmentos a X-Ray.
- BatchGetTraces: recupera una lista de rastros en una lista de ID de rastros. Cada rastro recuperado es una recopilación de documentos de segmentos de una única solicitud.
- GetTraceSummaries: recupera los ID y las anotaciones de los rastros. Puede especificar un `FilterExpression` para recuperar un subconjunto de resúmenes de rastreo.
- GetTraceGraph: recupera un gráfico de servicio para un ID de rastro específico.
- GetServiceGraph: recupera un documento con formato JSON que describe los servicios que procesan las solicitudes entrantes y las solicitudes posteriores a la llamada.

También puede usar la AWS Command Line Interface (AWS CLI) dentro del código de su aplicación para interactuar mediante programación con X-Ray. La AWS CLI es compatible con todas las funciones disponibles en el SDK de X-Ray, incluidas las de otros Servicios de AWS. Las siguientes funciones son versiones de las operaciones de la API indicadas anteriormente, pero con un formato más simple:

- put-trace-segments: carga documentos de segmentos a X-Ray.

- batch-get-traces: recupera una lista de rastros en una lista de ID de rastros. Cada rastro recuperado es una recopilación de documentos de segmentos de una única solicitud.
- get-trace-summaries: recupera los ID y las anotaciones de los rastros. Puede especificar un `FilterExpression` para recuperar un subconjunto de resúmenes de rastreo.
- get-trace-graph: recupera un gráfico de servicio para un ID de rastro específico.
- get-service-graph: recupera un documento con formato `JSON` que describe los servicios que procesan las solicitudes entrantes y las solicitudes posteriores a la llamada.

Para empezar, debe instalar la AWS CLI para su sistema operativo. AWS es compatible con sistemas operativos Linux, macOS y Windows. Para obtener más información sobre la lista de comandos de X-Ray, consulte la AWS CLI Command Reference guide for X-Ray.

Temas

- Uso de la API de AWS X-Ray con la CLI de AWS
- Envío de datos de seguimiento a AWS X-Ray
- Obtención de datos de AWS X-Ray
- Configuración de las opciones de muestreo, grupos y cifrado con la API de AWS X-Ray
- Using sampling rules with the X-Ray API (Uso de reglas de muestreo con la API de X-Ray)
- AWS X-RayDocumentos de segmentos de

Uso de la API de AWS X-Ray con la CLI de AWS

La interfaz de línea de comandos (CLI) de AWS le permite acceder al servicio de X-Ray directamente y utilizar las mismas API que la consola de X-Ray utiliza para recuperar los gráficos de servicios y los datos de rastro sin procesar. La aplicación de ejemplo incluye scripts que muestran cómo utilizar estas API con la CLI de AWS.

Requisitos previos

Este tutorial utiliza la aplicación de ejemplo Scorekeep e incluye scripts para generar datos de rastreo y un mapa de servicio. Siga las instrucciones en el tutorial de introducción para iniciar la aplicación.

Este tutorial utiliza la AWS CLI para mostrar el uso básico de la API de X-Ray. La CLI de AWS, disponible para Windows, Linux y OS-X, proporciona acceso de línea de comandos a las API públicas de todos los Servicios de AWS.

> **Note**
>
> Debe verificar que la AWS CLI esté configurada en la misma región en la que se creó la aplicación de ejemplo Scorekeep.

Los scripts incluidos para probar la aplicación de ejemplo utiliza `cURL` para enviar tráfico a la API y `jq` para analizar la salida. Puede descargar el ejecutable de `jq` desde stedolan.github.io y el ejecutable de `curl` desde https://curl.haxx.se/download.html. La mayoría de las instalaciones en Linux y OS X instalaciones incluyen cURL.

Generación de datos de rastreo

La aplicación web sigue funcionando para generar tráfico a la API cada pocos segundos mientras que la instalación está en curso, pero solo genera un tipo de solicitud. Utilice el script `test-api.sh` para ejecutar escenarios de un extremo a otro y generar datos de rastreo más diversos mientras prueba la API.

Para usar el script **`test-api.sh`**

1. Abra la consola de Elastic Beanstalk.
2. Desplácese hasta la consola de administración del entorno.
3. Copie la URL del entorno del encabezado de la página.
4. Abra el script `bin/test-api.sh` y reemplace el valor de la API con el URL del entorno.

   ```
   #!/bin/bash
   API=scorekeep.9hbtbm23t2.us-west-2.elasticbeanstalk.com/api
   ```

5. Ejecute el script para generar tráfico a la API.

   ```
   ~/debugger-tutorial$ ./bin/test-api.sh
   Creating users,
   session,
   game,
   configuring game,
   playing game,
   ending game,
   game complete.
   {"id":"MTBP8BAS","session":"HUF6IT64","name":"tic-tac-toe-test","users":
   ["QFF3HBGM","KL6JR98D"],"rules":"102","startTime":1476314241,"endTime":1476314245,"states":
   ```

```
["JQVLEOM2","D67QLPIC","VF9BM9NC","OEAA6GK9","2A705O73","1U2LFTLJ","HUKIDD70","BAN1C8FI","G
["BS8F8LQ","4MTTSPKP","463OETES","SVEBCL3N","N7CQ1GHP","O84ONEPD","EG4BPROQ","V4BLIDJ3","9R
```

Uso de la API de X-Ray

La CLI de AWS proporciona comandos para todas las acciones de API que X-Ray ofrece, incluidos los comandos `GetServiceGraph` y `GetTraceSummaries`. Consulte la Referencia de la API de AWS X-Ray para obtener más información acerca de todas las acciones admitidas y los tipos de datos que utilizan.

Example bin/service-graph.sh

```
EPOCH=$(date +%s)
aws xray get-service-graph --start-time $(($EPOCH-600)) --end-time $EPOCH
```

El script recupera un gráfico de servicios de los últimos 10 minutos.

```
~/eb-java-scorekeep$ ./bin/service-graph.sh | less
{
    "StartTime": 1479068648.0,
    "Services": [
        {
            "StartTime": 1479068648.0,
            "ReferenceId": 0,
            "State": "unknown",
            "EndTime": 1479068651.0,
            "Type": "client",
            "Edges": [
                {
                    "StartTime": 1479068648.0,
                    "ReferenceId": 1,
                    "SummaryStatistics": {
                        "ErrorStatistics": {
                            "ThrottleCount": 0,
                            "TotalCount": 0,
                            "OtherCount": 0
                        },
                        "FaultStatistics": {
                            "TotalCount": 0,
                            "OtherCount": 0
                        },
```

```
                        "TotalCount": 2,
                        "OkCount": 2,
                        "TotalResponseTime": 0.054000139236450195
                    },
                    "EndTime": 1479068651.0,
                    "Aliases": []
                }
            ]
        },
        {
            "StartTime": 1479068648.0,
            "Names": [
                "scorekeep.elasticbeanstalk.com"
            ],
            "ReferenceId": 1,
            "State": "active",
            "EndTime": 1479068651.0,
            "Root": true,
            "Name": "scorekeep.elasticbeanstalk.com",
...
```

Example bin/trace-urls.sh

```
EPOCH=$(date +%s)
aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time $(($EPOCH-60)) --
query 'TraceSummaries[*].Http.HttpURL'
```

El script recupera las URL de rastros generados hace uno o dos minutos.

```
~/eb-java-scorekeep$ ./bin/trace-urls.sh
[
    "http://scorekeep.elasticbeanstalk.com/api/game/6Q0UE1DG/5FGLM9U3/
endtime/1479069438",
    "http://scorekeep.elasticbeanstalk.com/api/session/KH4341QH",
    "http://scorekeep.elasticbeanstalk.com/api/game/GLQBJ3K5/153AHDIA",
    "http://scorekeep.elasticbeanstalk.com/api/game/VPDL672J/G2V41HM6/
endtime/1479069466"
]
```

Example bin/full-traces.sh

```
EPOCH=$(date +%s)
```

```
TRACEIDS=$(aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time
 $(($EPOCH-60)) --query 'TraceSummaries[*].Id' --output text)
aws xray batch-get-traces --trace-ids $TRACEIDS --query 'Traces[*]'
```

El script recupera los rastros completos generados hace uno o dos minutos.

```
~/eb-java-scorekeep$ ./bin/full-traces.sh | less
[
    {
        "Segments": [
            {
                "Id": "3f212bc237bafd5d",
                "Document": "{\"id\":\"3f212bc237bafd5d\",\"name\":\"DynamoDB\",
\"trace_id\":\"1-5828d9f2-a90669393f4343211bc1cf75\",\"start_time\":1.479072242459E9,
\"end_time\":1.479072242477E9,\"parent_id\":\"72a08dcf87991ca9\",\"http\":
{\"response\":{\"content_length\":60,\"status\":200}},\"inferred\":true,\"aws\":
{\"consistent_read\":false,\"table_name\":\"scorekeep-session-xray\",\"operation\":
\"GetItem\",\"request_id\":\"QAKE0S8DD0LJM245KAOPMA746BVV4KQNSO5AEMVJF66Q9ASUAAJG\",
\"resource_names\":[\"scorekeep-session-xray\"]},\"origin\":\"AWS::DynamoDB::Table\"}"
            },
            {
                "Id": "309e355f1148347f",
                "Document": "{\"id\":\"309e355f1148347f\",\"name\":\"DynamoDB\",
\"trace_id\":\"1-5828d9f2-a90669393f4343211bc1cf75\",\"start_time\":1.479072242477E9,
\"end_time\":1.479072242494E9,\"parent_id\":\"37f14ef837f00022\",\"http\":
{\"response\":{\"content_length\":606,\"status\":200}},\"inferred\":true,\"aws\":
{\"table_name\":\"scorekeep-game-xray\",\"operation\":\"UpdateItem\",\"request_id
\":\"388GEROC4PCA6D59ED3CTI5EEJVV4KQNSO5AEMVJF66Q9ASUAAJG\",\"resource_names\":
[\"scorekeep-game-xray\"]},\"origin\":\"AWS::DynamoDB::Table\"}"
            }
        ],
        "Id": "1-5828d9f2-a90669393f4343211bc1cf75",
        "Duration": 0.05099987983703613
    }
...
```

Eliminación

Finalice el entorno de Elastic Beanstalk para desactivar las instancias de Amazon EC2, las tablas de DynamoDB y otros recursos.

Para terminar su entorno de Elastic Beanstalk

1. Abra la consola de Elastic Beanstalk.
2. Desplácese hasta la consola de administración del entorno.
3. Elija Acciones.
4. Elija Terminate Environment (Terminar entorno).
5. Elija Finalizar.

Los datos de rastro se eliminan automáticamente de X-Ray después de 30 días.

Envío de datos de seguimiento a AWS X-Ray

Puede enviar datos de rastro a X-Ray en forma de documentos de segmento. Un documento de segmento es una cadena con formato JSON que contiene información sobre el trabajo que su aplicación realiza en respuesta a una solicitud. La aplicación puede registrar en segmentos los datos sobre el trabajo que realiza o bien registrar en subsegmentos los datos sobre el trabajo que utiliza servicios y recursos posteriores.

Los segmentos contienen información sobre el trabajo que realiza su aplicación. Un segmento, como mínimo, registra el tiempo empleado en una tarea, un nombre y dos ID. El ID de rastro permite controlar la solicitud mientras va de un servicio a otro. El ID de segmento permite controlar el trabajo que realiza un único servicio para la solicitud.

Example Segmento completo mínimo

```
{
  "name" : "Scorekeep",
  "id" : "70de5b6f19ff9a0a",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  "end_time" : 1.478293361449E9
}
```

Cuando se recibe una solicitud, puede enviar un segmento en curso como marcador hasta que se complete la solicitud.

Example Segmento en curso

```
{
```

```
  "name" : "Scorekeep",
  "id" : "70de5b6f19ff9a0b",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  “in_progress”: true
}
```

Puede enviar los segmentos a X-Ray directamente con `PutTraceSegments` o a través del daemon de X-Ray.

La mayoría de las aplicaciones llaman a otros servicios u obtienen acceso a los recursos con el SDK de AWS. Registre información acerca de las llamadas posteriores en subsegmentos. X-Ray utiliza subsegmentos para identificar los servicios posteriores que no envían segmentos y crean entradas para segmentos en el gráfico de servicios.

Un subsegmento puede incrustarse en un documento de segmentos completos o enviarse por separado. Envíe subsegmentos por separado para realizar un rastreo asíncrono de las llamadas posteriores para realizar solicitudes de larga ejecución o para evitar superar el tamaño máximo del documento de segmentos (64 kB).

Example Subsegmento

Un subsegmento tiene un `type` de `subsegment` y una `parent_id` que identifica el segmento de origen.

```
{
  "name" : "www2.example.com",
  "id" : "70de5b6f19ff9a0c",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979"
  “end_time” : 1.478293361449E9,
  “type” : “subsegment”,
  “parent_id” : “70de5b6f19ff9a0b”
}
```

Para obtener más información acerca de los campos y valores que puede incluir en los segmentos y subsegmentos, consulte AWS X-RayDocumentos de segmentos de.

Secciones

- Generación de identificadores de seguimiento
- Uso de PutTraceSegments

- Envío de documentos de segmento al daemon de X-Ray

Generación de identificadores de seguimiento

Para enviar datos a X-Ray, debe generar un ID de rastro único para cada solicitud.

Formato de ID de rastro de X-Ray

Un `trace_id` de X-Ray consta de tres números separados por guiones. Por ejemplo, `1-58406520-a006649127e371903a2de979`. Esto incluye:

- El número de versión, que es 1.
- La hora en que se realizó la solicitud original, en formato de tiempo Unix, en 8 dígitos hexadecimales.

  Por ejemplo, las 10:00 del 1 de diciembre de 2016, PST en formato de tiempo Unix es `1480615200` segundos o `58406520` en formato hexadecimal.
- Un identificador 96 bits del rastro, único a nivel global, en 24 dígitos hexadecimales.

> **Note**
>
> Ahora X-Ray admite ID de rastro creados mediante OpenTelemetry o cualquier otro marco que se ajuste a la especificación Trace Context de W3C. Un ID de rastro conforme a la especificación de W3C debe estar formateado en el formato de ID de rastro de X-Ray cuando se envía a X-Ray. Por ejemplo, el ID de rastro `4efaaf4d1e8720b39541901950019ee5` conforme a la especificación de W3C debe tener el formato `1-4efaaf4d-1e8720b39541901950019ee5` cuando se envíe a X-Ray. Si bien los ID de rastro de X-Ray incluyen la marca de tiempo de la solicitud original en formato de tiempo Unix, no es obligatorio cuando se envían ID de rastro conforme a la especificación de W3C en el formato de X-Ray.

Puede escribir un script con el fin de generar ID de rastro para pruebas. A continuación se incluyen dos ejemplos.

Python

```
import time
```

```
import os
import binascii

START_TIME = time.time()
HEX=hex(int(START_TIME))[2:]
TRACE_ID="1-{}-{}".format(HEX, binascii.hexlify(os.urandom(12)).decode('utf-8'))
```

Bash

```
START_TIME=$(date +%s)
HEX_TIME=$(printf '%x\n' $START_TIME)
GUID=$(dd if=/dev/random bs=12 count=1 2>/dev/null | od -An -tx1 | tr -d ' \t\n')
TRACE_ID="1-$HEX_TIME-$GUID"
```

Consulte la aplicación de muestra de Scorekeep para scripts que crean ID de rastro y envían segmentos al daemon de X-Ray.

- Python – xray_start.py
- Bash: xray_start.sh

Uso de PutTraceSegments

Puede cargar documentos de segmento con la API de PutTraceSegments. La API tiene un único parámetro, `TraceSegmentDocuments`, que obtiene una lista de documentos de segmento JSON.

Con la CLI de AWS, utilice el comando `aws xray put-trace-segments` para enviar documentos de segmento directamente a X-Ray.

```
$ DOC='{"trace_id": "1-5960082b-ab52431b496add878434aa25", "id": "6226467e3f845502",
 "start_time": 1498082657.37518, "end_time": 1498082695.4042, "name":
 "test.elasticbeanstalk.com"}'
$ aws xray put-trace-segments --trace-segment-documents "$DOC"
{
    "UnprocessedTraceSegments": []
}
```

> **Note**
>
> El procesador de comandos de Windows y Windows PowerShell tienen requisitos diferentes para añadir comillas o comillas con caracteres de escape en cadenas JSON. Consulte

Entrecomillado de cadenas en la Guía del usuario de la AWS CLI para obtener más información.

En la salida se indican los segmentos que no se han procesado correctamente; por ejemplo, si la fecha del ID de rastro es demasiado antigua, verá un error como el siguiente.

```
{
    "UnprocessedTraceSegments": [
        {
            "ErrorCode": "InvalidTraceId",
            "Message": "Invalid segment. ErrorCode: InvalidTraceId",
            "Id": "6226467e3f845502"
        }
    ]
}
```

Puede transferir varios documentos de segmento al mismo tiempo, separados por espacios.

```
$ aws xray put-trace-segments --trace-segment-documents "$DOC1" "$DOC2"
```

Envío de documentos de segmento al daemon de X-Ray

En lugar de enviar documentos de segmento a la API de X-Ray, puede enviar segmentos y subsegmentos al daemon de X-Ray, que se encargará de almacenarlos en búfer y cargarlos en la API de X-Ray en lotes. El SDK de X-Ray envía documentos de segmento al daemon para evitar llamar directamente a AWS.

Note

Consulte Ejecución del daemon de X-Ray localmente para obtener instrucciones sobre cómo ejecutar el demonio.

Envíe el segmento en formato JSON a través del puerto UDP 2000, anteponiéndole el encabezado del demonio, `{"format": "json", "version": 1}\n`

```
{"format": "json", "version": 1}\n{"trace_id": "1-5759e988-bd862e3fe1be46a994272793",
 "id": "defdfd9912dc5a56", "start_time": 1461096053.37518, "end_time": 1461096053.4042,
 "name": "test.elasticbeanstalk.com"}
```

En Linux, puede enviar documentos de segmento al demonio desde un terminal Bash. Guarde el encabezado y el documento de segmento en un archivo de texto y envíelo a `/dev/udp` con `cat`.

```
$ cat segment.txt > /dev/udp/127.0.0.1/2000
```

Example segment.txt

```
{"format": "json", "version": 1}
{"trace_id": "1-594aed87-ad72e26896b3f9d3a27054bb", "id": "6226467e3f845502",
 "start_time": 1498082657.37518, "end_time": 1498082695.4042, "name":
 "test.elasticbeanstalk.com"}
```

Consulte el registro del daemon para comprobar que ha enviado el segmento a X-Ray.

```
2017-07-07T01:57:24Z [Debug] processor: sending partial batch
2017-07-07T01:57:24Z [Debug] processor: segment batch size: 1. capacity: 50
2017-07-07T01:57:24Z [Info] Successfully sent batch of 1 segments (0.020 seconds)
```

Obtención de datos de AWS X-Ray

AWS X-Ray procesa los datos de rastreo que usted le envía para generar registros de seguimiento completos, resúmenes de registros de seguimiento y gráficos de servicios en JSON. Puede recuperar los datos generados directamente desde la API con la CLI de AWS.

Secciones

- Recuperación del gráfico de servicios
- Recuperación del gráfico de servicios por grupo
- Recuperación de registros de seguimiento
- Recuperación y ajuste de las causas raíz de Analytics

Recuperación del gráfico de servicios

Puede utilizar la API de `GetServiceGraph` para recuperar el gráfico de servicios de JSON. La API requiere una hora de inicio y de finalización, que se puede calcular a partir de un terminal de Linux con el comando `date`.

```
$ date +%s
1499394617
```

`date +%s` imprime una fecha en cuestión de segundos. Utilice este número como hora de finalización y réstele el tiempo para obtener una hora de inicio.

Example Script para recuperar un gráfico de servicios de los últimos 10 minutos

```
EPOCH=$(date +%s)
aws xray get-service-graph --start-time $(($EPOCH-600)) --end-time $EPOCH
```

En el siguiente ejemplo, se muestra un gráfico de servicio con cuatro nodos que incluye un nodo cliente, una instancia de EC2, una tabla de DynamoDB y un tema de Amazon SNS.

Example Salida de GetServiceGraph

```
{
    "Services": [
        {
            "ReferenceId": 0,
            "Name": "xray-sample.elasticbeanstalk.com",
            "Names": [
                "xray-sample.elasticbeanstalk.com"
            ],
            "Type": "client",
            "State": "unknown",
            "StartTime": 1528317567.0,
            "EndTime": 1528317589.0,
            "Edges": [
                {
                    "ReferenceId": 2,
                    "StartTime": 1528317567.0,
                    "EndTime": 1528317589.0,
                    "SummaryStatistics": {
                        "OkCount": 3,
                        "ErrorStatistics": {
```

```
                        "ThrottleCount": 0,
                        "OtherCount": 1,
                        "TotalCount": 1
                    },
                    "FaultStatistics": {
                        "OtherCount": 0,
                        "TotalCount": 0
                    },
                    "TotalCount": 4,
                    "TotalResponseTime": 0.273
                },
                "ResponseTimeHistogram": [
                    {
                        "Value": 0.005,
                        "Count": 1
                    },
                    {
                        "Value": 0.015,
                        "Count": 1
                    },
                    {
                        "Value": 0.157,
                        "Count": 1
                    },
                    {
                        "Value": 0.096,
                        "Count": 1
                    }
                ],
                "Aliases": []
            }
        ]
    },
    {
        "ReferenceId": 1,
        "Name": "awseb-e-dixzws4s9p-stack-StartupSignupsTable-4IMSMHAYX2BA",
        "Names": [
            "awseb-e-dixzws4s9p-stack-StartupSignupsTable-4IMSMHAYX2BA"
        ],
        "Type": "AWS::DynamoDB::Table",
        "State": "unknown",
        "StartTime": 1528317583.0,
        "EndTime": 1528317589.0,
        "Edges": [],
```

```
            "SummaryStatistics": {
                "OkCount": 2,
                "ErrorStatistics": {
                    "ThrottleCount": 0,
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "FaultStatistics": {
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "TotalCount": 2,
                "TotalResponseTime": 0.12
            },
            "DurationHistogram": [
                {
                    "Value": 0.076,
                    "Count": 1
                },
                {
                    "Value": 0.044,
                    "Count": 1
                }
            ],
            "ResponseTimeHistogram": [
                {
                    "Value": 0.076,
                    "Count": 1
                },
                {
                    "Value": 0.044,
                    "Count": 1
                }
            ]
        },
        {
            "ReferenceId": 2,
            "Name": "xray-sample.elasticbeanstalk.com",
            "Names": [
                "xray-sample.elasticbeanstalk.com"
            ],
            "Root": true,
            "Type": "AWS::EC2::Instance",
            "State": "active",
```

```
            "StartTime": 1528317567.0,
            "EndTime": 1528317589.0,
            "Edges": [
                {
                    "ReferenceId": 1,
                    "StartTime": 1528317567.0,
                    "EndTime": 1528317589.0,
                    "SummaryStatistics": {
                        "OkCount": 2,
                        "ErrorStatistics": {
                            "ThrottleCount": 0,
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
                        "FaultStatistics": {
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
                        "TotalCount": 2,
                        "TotalResponseTime": 0.12
                    },
                    "ResponseTimeHistogram": [
                        {
                            "Value": 0.076,
                            "Count": 1
                        },
                        {
                            "Value": 0.044,
                            "Count": 1
                        }
                    ],
                    "Aliases": []
                },
                {
                    "ReferenceId": 3,
                    "StartTime": 1528317567.0,
                    "EndTime": 1528317589.0,
                    "SummaryStatistics": {
                        "OkCount": 2,
                        "ErrorStatistics": {
                            "ThrottleCount": 0,
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
```

```
                    "FaultStatistics": {
                        "OtherCount": 0,
                        "TotalCount": 0
                    },
                    "TotalCount": 2,
                    "TotalResponseTime": 0.125
                },
                "ResponseTimeHistogram": [
                    {
                        "Value": 0.049,
                        "Count": 1
                    },
                    {
                        "Value": 0.076,
                        "Count": 1
                    }
                ],
                "Aliases": []
            }
        ],
        "SummaryStatistics": {
            "OkCount": 3,
            "ErrorStatistics": {
                "ThrottleCount": 0,
                "OtherCount": 1,
                "TotalCount": 1
            },
            "FaultStatistics": {
                "OtherCount": 0,
                "TotalCount": 0
            },
            "TotalCount": 4,
            "TotalResponseTime": 0.273
        },
        "DurationHistogram": [
            {
                "Value": 0.005,
                "Count": 1
            },
            {
                "Value": 0.015,
                "Count": 1
            },
            {
```

```
                    "Value": 0.157,
                    "Count": 1
                },
                {
                    "Value": 0.096,
                    "Count": 1
                }
            ],
            "ResponseTimeHistogram": [
                {
                    "Value": 0.005,
                    "Count": 1
                },
                {
                    "Value": 0.015,
                    "Count": 1
                },
                {
                    "Value": 0.157,
                    "Count": 1
                },
                {
                    "Value": 0.096,
                    "Count": 1
                }
            ]
        },
        {
            "ReferenceId": 3,
            "Name": "SNS",
            "Names": [
                "SNS"
            ],
            "Type": "AWS::SNS",
            "State": "unknown",
            "StartTime": 1528317583.0,
            "EndTime": 1528317589.0,
            "Edges": [],
            "SummaryStatistics": {
                "OkCount": 2,
                "ErrorStatistics": {
                    "ThrottleCount": 0,
                    "OtherCount": 0,
                    "TotalCount": 0
```

```
                },
                "FaultStatistics": {
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "TotalCount": 2,
                "TotalResponseTime": 0.125
            },
            "DurationHistogram": [
                {
                    "Value": 0.049,
                    "Count": 1
                },
                {
                    "Value": 0.076,
                    "Count": 1
                }
            ],
            "ResponseTimeHistogram": [
                {
                    "Value": 0.049,
                    "Count": 1
                },
                {
                    "Value": 0.076,
                    "Count": 1
                }
            ]
        }
    ]
}
```

Recuperación del gráfico de servicios por grupo

Para obtener un gráfico de servicios en función del contenido de un grupo, incluya un `groupName` o `groupARN`. En el ejemplo siguiente, se muestra una llamada a un gráfico de servicios para un grupo denominado Example1.

Example Script para recuperar un gráfico de servicios por nombre para el grupo Example1

```
aws xray get-service-graph --group-name "Example1"
```

Recuperación de registros de seguimiento

Puede utilizar la API de `GetTraceSummaries` para obtener una lista de resúmenes de rastros. Estos resúmenes incluyen información que puede utilizar para identificar los registros de seguimiento que desee descargar en su totalidad, y que incluyen anotaciones, información de solicitud y respuesta e identificadores.

Existen dos indicadores `TimeRangeType` disponibles al llamar `aws xray get-trace-summaries`:

- TraceId: la búsqueda predeterminada con `GetTraceSummaries` utiliza el tiempo de TraceID y devuelve los registros de seguimiento iniciados dentro del intervalo `[start_time, end_time)` calculado. Este rango de marcas de tiempo se calcula en función de la codificación de la marca de tiempo en TraceId o se puede definir de forma manual.
- Hora del evento: para buscar eventos a medida que ocurren a lo largo del tiempo, AWS X-Ray permite buscar registros de seguimiento mediante marcas de tiempo de eventos. El tiempo de evento devuelve registros de seguimiento activos durante el intervalo `[start_time, end_time)`, independientemente del cuándo se inició el seguimiento.

Utilice el comando `aws xray get-trace-summaries` para obtener una lista de resúmenes de registros de seguimiento. Los siguientes comandos le permiten obtener una lista de resúmenes de registros de seguimiento con una antigüedad de entre 1 y 2 minutos mediante el tiempo predeterminado de TraceId.

Example Script para obtener resúmenes de rastreo

```
EPOCH=$(date +%s)
aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time $(($EPOCH-60))
```

Example Salida de GetTraceSummaries

```
{
    "TraceSummaries": [
        {
            "HasError": false,
            "Http": {
                "HttpStatus": 200,
                "ClientIp": "205.255.255.183",
                "HttpURL": "http://scorekeep.elasticbeanstalk.com/api/session",
```

```
                "UserAgent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36",
                "HttpMethod": "POST"
            },
            "Users": [],
            "HasFault": false,
            "Annotations": {},
            "ResponseTime": 0.084,
            "Duration": 0.084,
            "Id": "1-59602606-a43a1ac52fc7ee0eea12a82c",
            "HasThrottle": false
        },
        {
            "HasError": false,
            "Http": {
                "HttpStatus": 200,
                "ClientIp": "205.255.255.183",
                "HttpURL": "http://scorekeep.elasticbeanstalk.com/api/user",
                "UserAgent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36",
                "HttpMethod": "POST"
            },
            "Users": [
                {
                    "UserName": "5M388M1E"
                }
            ],
            "HasFault": false,
            "Annotations": {
                "UserID": [
                    {
                        "AnnotationValue": {
                            "StringValue": "5M388M1E"
                        }
                    }
                ],
                "Name": [
                    {
                        "AnnotationValue": {
                            "StringValue": "Ola"
                        }
                    }
                ]
            },
```

```
            "ResponseTime": 3.232,
            "Duration": 3.232,
            "Id": "1-59602603-23fc5b688855d396af79b496",
            "HasThrottle": false
        }
    ],
    "ApproximateTime": 1499473304.0,
    "TracesProcessedCount": 2
}
```

Utilice el ID de rastro del resultado para recuperar un registro de seguimiento completo con la API de `BatchGetTraces`.

Example Comando BatchGetTraces

```
$ aws xray batch-get-traces --trace-ids 1-596025b4-7170afe49f7aa708b1dd4a6b
```

Example Salida de BatchGetTraces

```
{
    "Traces": [
        {
            "Duration": 3.232,
            "Segments": [
                {
                    "Document": "{\"id\":\"1fb07842d944e714\",\"name\":
\"random-name\",\"start_time\":1.499473411677E9,\"end_time\":1.499473414572E9,
\"parent_id\":\"0c544c1b1bbff948\",\"http\":{\"response\":{\"status\":200}},
\"aws\":{\"request_id\":\"ac086670-6373-11e7-a174-f31b3397f190\"},\"trace_id\":
\"1-59602603-23fc5b688855d396af79b496\",\"origin\":\"AWS::Lambda\",\"resource_arn\":
\"arn:aws:lambda:us-west-2:123456789012:function:random-name\"}",
                    "Id": "1fb07842d944e714"
                },
                {
                    "Document": "{\"id\":\"194fcc8747581230\",\"name\":\"Scorekeep
\",\"start_time\":1.499473411562E9,\"end_time\":1.499473414794E9,\"http\":{\"request
\":{\"url\":\"http://scorekeep.elasticbeanstalk.com/api/user\",\"method\":\"POST\",
\"user_agent\":\"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/59.0.3071.115 Safari/537.36\",\"client_ip\":\"205.251.233.183\"},
\"response\":{\"status\":200}},\"aws\":{\"elastic_beanstalk\":{\"version_label\":\"app-
abb9-170708_002045\",\"deployment_id\":406,\"environment_name\":\"scorekeep-dev\"},
\"ec2\":{\"availability_zone\":\"us-west-2c\",\"instance_id\":\"i-0cd9e448944061b4a
\"},\"xray\":{\"sdk_version\":\"1.1.2\",\"sdk\":\"X-Ray for Java\"}},\"service
```

\":{},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"user\":\"5M388M1E
\",\"origin\":\"AWS::ElasticBeanstalk::Environment\",\"subsegments\":[{\"id\":
\"0c544c1b1bbff948\",\"name\":\"Lambda\",\"start_time\":1.499473411629E9,\"end_time
\":1.499473414572E9,\"http\":{\"response\":{\"status\":200,\"content_length\":14}},
\"aws\":{\"log_type\":\"None\",\"status_code\":200,\"function_name\":\"random-name
\",\"invocation_type\":\"RequestResponse\",\"operation\":\"Invoke\",\"request_id
\":\"ac086670-6373-11e7-a174-f31b3397f190\",\"resource_names\":[\"random-name\"]},
\"namespace\":\"aws\"},{\"id\":\"071684f2e555e571\",\"name\":\"## UserModel.saveUser
\",\"start_time\":1.499473414581E9,\"end_time\":1.499473414769E9,\"metadata\":{\"debug
\":{\"test\":\"Metadata string from UserModel.saveUser\"}},\"subsegments\":[{\"id\":
\"4cd3f10b76c624b4\",\"name\":\"DynamoDB\",\"start_time\":1.49947341469E9,\"end_time
\":1.499473414769E9,\"http\":{\"response\":{\"status\":200,\"content_length\":57}},
\"aws\":{\"table_name\":\"scorekeep-user\",\"operation\":\"UpdateItem\",\"request_id
\":\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738BOB4KQNSO5AEMVJF66Q9\",\"resource_names\":
[\"scorekeep-user\"]},\"namespace\":\"aws\"}]}]}",
                    "Id": "194fcc8747581230"
                },
                {
                    "Document": "{\"id\":\"00f91aa01f4984fd\",\"name\":
\"random-name\",\"start_time\":1.49947341283E9,\"end_time\":1.49947341457E9,
\"parent_id\":\"1fb07842d944e714\",\"aws\":{\"function_arn\":\"arn:aws:lambda:us-
west-2:123456789012:function:random-name\",\"resource_names\":[\"random-name\"],
\"account_id\":\"123456789012\"},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",
\"origin\":\"AWS::Lambda::Function\",\"subsegments\":[{\"id\":\"e6d2fe619f827804\",
\"name\":\"annotations\",\"start_time\":1.499473413012E9,\"end_time\":1.499473413069E9,
\"annotations\":{\"UserID\":\"5M388M1E\",\"Name\":\"Ola\"}},{\"id\":\"b29b548af4d54a0f
\",\"name\":\"SNS\",\"start_time\":1.499473413112E9,\"end_time\":1.499473414071E9,
\"http\":{\"response\":{\"status\":200}},\"aws\":{\"operation\":\"Publish\",
\"region\":\"us-west-2\",\"request_id\":\"a2137970-f6fc-5029-83e8-28aadeb99198\",
\"retries\":0,\"topic_arn\":\"arn:aws:sns:us-west-2:123456789012:awseb-e-
ruag3jyweb-stack-NotificationTopic-6B829NT9V5O9\"},\"namespace\":\"aws\"},{\"id\":
\"2279c0030c955e52\",\"name\":\"Initialization\",\"start_time\":1.499473412064E9,
\"end_time\":1.499473412819E9,\"aws\":{\"function_arn\":\"arn:aws:lambda:us-
west-2:123456789012:function:random-name\"}}]}",
                    "Id": "00f91aa01f4984fd"
                },
                {
                    "Document": "{\"id\":\"17ba309b32c7fbaf\",\"name\":
\"DynamoDB\",\"start_time\":1.49947341469E9,\"end_time\":1.499473414769E9,
\"parent_id\":\"4cd3f10b76c624b4\",\"inferred\":true,\"http\":{\"response
\":{\"status\":200,\"content_length\":57}},\"aws\":{\"table_name
\":\"scorekeep-user\",\"operation\":\"UpdateItem\",\"request_id\":
\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738BOB4KQNSO5AEMVJF66Q9\",\"resource_names\":

```
[\"scorekeep-user\"]},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"origin\":
\"AWS::DynamoDB::Table\"}",
                    "Id": "17ba309b32c7fbaf"
                },
                {
                    "Document": "{\"id\":\"1ee3c4a523f89ca5\",\"name\":\"SNS
\",\"start_time\":1.499473413112E9,\"end_time\":1.499473414071E9,\"parent_id\":
\"b29b548af4d54a0f\",\"inferred\":true,\"http\":{\"response\":{\"status\":200}},\"aws
\":{\"operation\":\"Publish\",\"region\":\"us-west-2\",\"request_id\":\"a2137970-
f6fc-5029-83e8-28aadeb99198\",\"retries\":0,\"topic_arn\":\"arn:aws:sns:us-
west-2:123456789012:awseb-e-ruag3jyweb-stack-NotificationTopic-6B829NT9V5O9\"},
\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"origin\":\"AWS::SNS\"}",
                    "Id": "1ee3c4a523f89ca5"
                }
            ],
            "Id": "1-59602603-23fc5b688855d396af79b496"
        }
    ],
    "UnprocessedTraceIds": []
}
```


El registro de seguimiento completo incluye un documento para cada segmento, compilado a partir de todos los documentos de segmento recibidos que tienen el mismo ID de registro de seguimiento. Estos documentos no representan los datos tal y como la aplicación se los envió a X-Ray, sino que representan los documentos procesados generados por el servicio de X-Ray. X-Ray crea el documento de rastro completo compilando los documentos de segmento que envía la aplicación, y eliminando los datos que no se ajustan al esquema de documentos de segmento.

X-Ray también crea segmentos inferidos para las llamadas posteriores a los servicios que no envían los segmentos propiamente dichos. Por ejemplo, si llama a DynamoDB con un cliente instrumentado, el SDK de X-Ray registra un subsegmento con detalles sobre la llamada desde su punto de vista, pero no envía el segmento correspondiente. X-Ray utiliza la información del subsegmento con el fin de crear un segmento inferido para representar el recurso de DynamoDB en el mapa de rastros y lo añade al documento de rastro.

Para obtener varios registros de seguimiento desde la API, se necesita una lista de ID de registros de seguimiento, que se puede extraer de la salida de `get-trace-summaries` con una consulta de la AWS CLI. Redirija la lista a la entrada de `batch-get-traces` para obtener registros de seguimiento completos de un período específico.

Example Script para obtener registros de seguimiento completos de un período de un minuto

```
EPOCH=$(date +%s)
TRACEIDS=$(aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time
 $(($EPOCH-60)) --query 'TraceSummaries[*].Id' --output text)
aws xray batch-get-traces --trace-ids $TRACEIDS --query 'Traces[*]'
```

Recuperación y ajuste de las causas raíz de Analytics

Al generar un resumen de registros de seguimiento con la API GetTraceSummaries. se pueden reutilizar los resúmenes de rastros parciales en formato JSON para crear una expresión de filtro más precisa en función de las causas raíz. Consulte los ejemplos que aparecen a continuación para ver el procedimiento de ajuste.

Example Ejemplo de salida de GetTraceSummaries: sección de causa raíz de tiempo de respuesta


```
{
  "Services": [
    {
      "Name": "GetWeatherData",
      "Names": ["GetWeatherData"],
      "AccountId": 123456789012,
      "Type": null,
      "Inferred": false,
      "EntityPath": [
        {
          "Name": "GetWeatherData",
          "Coverage": 1.0,
          'Remote": false
        },
        {
          "Name": "get_temperature",
          "Coverage": 0.8,
          "Remote": false
        }
      ]
    },
    {
      "Name": "GetTemperature",
      "Names": ["GetTemperature"],
      "AccountId": 123456789012,
      "Type": null,
      "Inferred": false,
```

```
      "EntityPath": [
        {
          "Name": "GetTemperature",
          "Coverage": 0.7,
          "Remote": false
        }
      ]
    }
  ]
}
```

Al editar y crear omisiones en la salida anterior, este JSON puede actuar de filtro para las entidades de causa raíz coincidentes. Para cada campo presente en el JSON, todas las coincidencias candidatas deben ser exactas o no se devolverá el registro de seguimiento. Los campos eliminados se convierten en valores comodín, un formato que es compatible con la estructura de consultas de expresión de filtro.

Example Causa raíz de tiempo de respuesta reformateado

```
{
  "Services": [
    {
      "Name": "GetWeatherData",
      "EntityPath": [
        {
          "Name": "GetWeatherData"
        },
        {
          "Name": "get_temperature"
        }
      ]
    },
    {
      "Name": "GetTemperature",
      "EntityPath": [
        {
          "Name": "GetTemperature"
        }
      ]
    }
  ]
}
```

Este JSON se utiliza como parte de una expresión de filtro a través de una llamada a `rootcause.json = #[{}]`. Consulte el capítulo Expresiones de filtro para obtener más información sobre cómo ejecutar consultas con expresiones de filtro.

Example Ejemplo de filtro JSON

```
rootcause.json = #[{ "Services": [ { "Name": "GetWeatherData", "EntityPath": [{ "Name":
 "GetWeatherData" }, { "Name": "get_temperature" } ] }, { "Name": "GetTemperature",
 "EntityPath": [ { "Name": "GetTemperature" } ] } ] }]
```

Configuración de las opciones de muestreo, grupos y cifrado con la API de AWS X-Ray

AWS X-Ray dispone de varias API para configurar las reglas de muestreo, las reglas de grupos y la configuración de cifrado.

Secciones

- Configuración de cifrado
- Reglas de muestreo
- Grupos

Configuración de cifrado

Se usa `PutEncryptionConfig` para especificar una clave de AWS Key Management Service (AWS KMS) que se usará para el cifrado.

> **Note**
>
> X-Ray no es compatible con claves de KMS asimétricas.

```
$ aws xray put-encryption-config --type KMS --key-id alias/aws/xray
{
    "EncryptionConfig": {
        "KeyId": "arn:aws:kms:us-east-2:123456789012:key/c234g4e8-39e9-4gb0-84e2-
b0ea215cbba5",
        "Status": "UPDATING",
        "Type": "KMS"
```

```
    }
}
```

Para el ID de clave, puede utilizar un alias (tal y como se muestra en el ejemplo), un ID de clave o un nombre de recurso de Amazon (ARN).

Utilice `GetEncryptionConfig` para obtener la configuración actual. Cuando X-Ray termina de aplicar la configuración, el estado cambia de `UPDATING` a `ACTIVE`.

```
$ aws xray get-encryption-config
{
    "EncryptionConfig": {
        "KeyId": "arn:aws:kms:us-east-2:123456789012:key/c234g4e8-39e9-4gb0-84e2-
b0ea215cbba5",
        "Status": "ACTIVE",
        "Type": "KMS"
    }
}
```

Para dejar de usar una clave de KMS y utilizar el cifrado predeterminado, establezca el tipo de cifrado en `NONE`.

```
$ aws xray put-encryption-config --type NONE
{
    "EncryptionConfig": {
        "Status": "UPDATING",
        "Type": "NONE"
    }
}
```

Reglas de muestreo

Puede administrar las reglas de muestreo en su cuenta con la API de X-Ray. Para obtener más información acerca de cómo añadir y administrar etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.

Obtener todas las reglas de muestreo con `GetSamplingRules`.

```
$ aws xray get-sampling-rules
{
    "SamplingRuleRecords": [
        {
```

```
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-2:123456789012:sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.05,
                "ReservoirSize": 1,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1529959993.0
        }
    ]
}
```

La regla predeterminada se aplica a todas las solicitudes que no coinciden con otra regla. Es la regla con prioridad más baja y no se puede eliminar. Sin embargo, puede cambiar el porcentaje y el tamaño de depósito con `UpdateSamplingRule`.

Example Entrada de la API para **`UpdateSamplingRule`** – 10000-default.json

```
{
    "SamplingRuleUpdate": {
        "RuleName": "Default",
        "FixedRate": 0.01,
        "ReservoirSize": 0
    }
}
```

En el siguiente ejemplo se utiliza el archivo anterior como entrada para cambiar la regla predeterminada a uno por ciento sin depósito. Las etiquetas son opcionales. Si decide añadir etiquetas, tenga en cuenta que le hará falta una clave de etiqueta y que los valores de etiqueta son opcionales. Para eliminar etiquetas existentes de una regla de muestreo, utilice UntagResource.

```
$ aws xray update-sampling-rule --cli-input-json file://1000-default.json --tags
 [{"Key": "key_name","Value": "value"},{"Key": "key_name","Value": "value"}]
```

```
{
    "SamplingRuleRecords": [
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-2:123456789012:sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.01,
                "ReservoirSize": 0,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1529959993.0
        },
```

Crear reglas de muestreo adicionales con `CreateSamplingRule`. Al crear una regla, la mayoría de los campos de la regla son obligatorios. En el siguiente ejemplo se crean dos reglas. Esta primera regla establece un porcentaje de base para la aplicación de ejemplo Scorekeep. Empareja todas las solicitudes servidas por la API que no coinciden con una regla de prioridad superior.

Example Entrada de la API para **`UpdateSamplingRule`** – 9000-base-scorekeep.json

```
{
    "SamplingRule": {
        "RuleName": "base-scorekeep",
        "ResourceARN": "*",
        "Priority": 9000,
        "FixedRate": 0.1,
        "ReservoirSize": 5,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1
    }
```

```
}
```

La segunda regla también se aplica a Scorekeep, pero tiene una prioridad mayor y es más específica. Esta regla establece un porcentaje de muestreo muy bajo para las solicitudes de sondeo. Estas son las solicitudes GET realizadas por el cliente cada pocos segundos para comprobar si hay cambios en el estado del juego.

Example Entrada de la API para **UpdateSamplingRule** – 5000-polling-scorekeep.json

```
{
    "SamplingRule": {
        "RuleName": "polling-scorekeep",
        "ResourceARN": "*",
        "Priority": 5000,
        "FixedRate": 0.003,
        "ReservoirSize": 0,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "GET",
        "URLPath": "/api/state/*",
        "Version": 1
    }
}
```

Las etiquetas son opcionales. Si decide añadir etiquetas, tenga en cuenta que le hará falta una clave de etiqueta y que los valores de etiqueta son opcionales.

```
$ aws xray create-sampling-rule --cli-input-json file://5000-polling-scorekeep.json --
tags [{"Key": "key_name","Value": "value"},{"Key": "key_name","Value": "value"}]
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "polling-scorekeep",
            "RuleARN": "arn:aws:xray:us-east-1:123456789012:sampling-rule/polling-
scorekeep",
            "ResourceARN": "*",
            "Priority": 5000,
            "FixedRate": 0.003,
            "ReservoirSize": 0,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
```

```
            "Host": "*",
            "HTTPMethod": "GET",
            "URLPath": "/api/state/*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574399.0,
        "ModifiedAt": 1530574399.0
    }
}
$ aws xray create-sampling-rule --cli-input-json file://9000-base-scorekeep.json
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "base-scorekeep",
            "RuleARN": "arn:aws:xray:us-east-1:123456789012:sampling-rule/base-
scorekeep",
            "ResourceARN": "*",
            "Priority": 9000,
            "FixedRate": 0.1,
            "ReservoirSize": 5,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "*",
            "URLPath": "*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574410.0,
        "ModifiedAt": 1530574410.0
    }
}
```

Para eliminar una regla de muestreo, utilice `DeleteSamplingRule`.

```
$ aws xray delete-sampling-rule --rule-name polling-scorekeep
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "polling-scorekeep",
            "RuleARN": "arn:aws:xray:us-east-1:123456789012:sampling-rule/polling-
scorekeep",
```

```
            "ResourceARN": "*",
            "Priority": 5000,
            "FixedRate": 0.003,
            "ReservoirSize": 0,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "GET",
            "URLPath": "/api/state/*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574399.0,
        "ModifiedAt": 1530574399.0
    }
}
```

Grupos

Puede utilizar la API de X-Ray para administrar los grupos de su cuenta. Los grupos son una colección de rastros que se definen mediante una expresión de filtro. Puede utilizar grupos para generar gráficos de servicios adicionales y suministrar métricas de Amazon CloudWatch. Consulte Obtención de datos de AWS X-Ray para obtener más detalles sobre cómo utilizar las métricas y los gráficos de servicios mediante la API de X-Ray. Para obtener más información acerca de los grupos, consulte Configuración de grupos. Para obtener más información acerca de cómo añadir y administrar etiquetas, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.

Utilice para crear un grupo `CreateGroup`. Las etiquetas son opcionales. Si decide añadir etiquetas, tenga en cuenta que le hará falta una clave de etiqueta y que los valores de etiqueta son opcionales.

```
$ aws xray create-group --group-name "TestGroup" --filter-expression
 "service(\"example.com\") {fault}" --tags [{"Key": "key_name","Value": "value"},
{"Key": "key_name","Value": "value"}]
{
    "GroupName": "TestGroup",
    "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup/UniqueID",
    "FilterExpression": "service(\"example.com\") {fault OR error}"
}
```

Utilice para obtener todos los grupos existentes `GetGroups`.

```
$ aws xray get-groups
```

```
{
    "Groups": [
        {
            "GroupName": "TestGroup",
            "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup/UniqueID",
            "FilterExpression": "service(\"example.com\") {fault OR error}"
        },
  {
            "GroupName": "TestGroup2",
            "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup2/
UniqueID",
            "FilterExpression": "responsetime > 2"
        }
    ],
 "NextToken": "tokenstring"
}
```

Utilice para actualizar un grupo `UpdateGroup`. Las etiquetas son opcionales. Si decide añadir etiquetas, tenga en cuenta que le hará falta una clave de etiqueta y que los valores de etiqueta son opcionales. Para eliminar etiquetas existentes de un grupo, utilice UntagResource.

```
$ aws xray update-group --group-name "TestGroup" --group-arn "arn:aws:xray:us-
east-2:123456789012:group/TestGroup/UniqueID" --filter-expression
 "service(\"example.com\") {fault OR error}" --tags [{"Key": "Stage","Value": "Prod"},
{"Key": "Department","Value": "QA"}]
{
    "GroupName": "TestGroup",
    "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup/UniqueID",
    "FilterExpression": "service(\"example.com\") {fault OR error}"
}
```

Utilice para eliminar un grupo `DeleteGroup`.

```
$ aws xray delete-group --group-name "TestGroup" --group-arn "arn:aws:xray:us-
east-2:123456789012:group/TestGroup/UniqueID"
    {
    }
```

Using sampling rules with the X-Ray API (Uso de reglas de muestreo con la API de X-Ray)

El SDK de AWS X-Ray utiliza la API de X-Ray para obtener reglas de muestreo, notificar resultados de muestreo y obtener las cuotas. Puede utilizar estas API para entender mejor cómo funcionan las reglas de muestreo o para implementar muestras en un lenguaje que el SDK de X-Ray no admita.

Comience por obtener todas las reglas de muestreo con `GetSamplingRules`.

```
$ aws xray get-sampling-rules
{
    "SamplingRuleRecords": [
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.01,
                "ReservoirSize": 0,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1530558121.0
        },
        {
            "SamplingRule": {
                "RuleName": "base-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/base-scorekeep",
                "ResourceARN": "*",
                "Priority": 9000,
                "FixedRate": 0.1,
                "ReservoirSize": 2,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
```

```
            },
            "CreatedAt": 1530573954.0,
            "ModifiedAt": 1530920505.0
        },
        {
            "SamplingRule": {
                "RuleName": "polling-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/polling-scorekeep",
                "ResourceARN": "*",
                "Priority": 5000,
                "FixedRate": 0.003,
                "ReservoirSize": 0,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "GET",
                "URLPath": "/api/state/*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530918163.0,
            "ModifiedAt": 1530918163.0
        }
    ]
}
```

El resultado incluye la regla predeterminada y las reglas personalizadas. Consulte Reglas de muestreo si aún no ha creado reglas de muestreo.

Evalúe las reglas frente a las solicitudes entrantes en orden de prioridad ascendente. Cuando una regla coincida, utilice el porcentaje fijo y el tamaño de depósito para tomar una decisión de muestreo. Registre solicitudes muestreadas y pase por alto (para fines de rastreo) las solicitudes sin muestrear. Deje de evaluar las reglas cuando tome una decisión de muestreo.

El tamaño de un depósito de reglas es el número objetivo de rastreos que registrar por segundo antes de aplicar el porcentaje fijo. El depósito se aplica en todos los servicios acumulativamente, por lo que no se puede utilizar directamente. Sin embargo, si es distinto de cero, podrá tomar prestado un rastro por segundo desde el depósito hasta que X-Ray asigne una cuota. Antes de recibir una cuota, registre la primera solicitud cada segundo y aplique el porcentaje fijo a las solicitudes adicionales. El porcentaje fijo es un número decimal entre 0 y 1,00 (100 %).

El siguiente ejemplo muestra una llamada a `GetSamplingTargets` con información detallada sobre las decisiones de muestreo tomadas durante los últimos 10 segundos.

```
$ aws xray get-sampling-targets --sampling-statistics-documents '[
    {
        "RuleName": "base-scorekeep",
        "ClientID": "ABCDEF1234567890ABCDEF10",
        "Timestamp": "2018-07-07T00:20:06",
        "RequestCount": 110,
        "SampledCount": 20,
        "BorrowCount": 10
    },
    {
        "RuleName": "polling-scorekeep",
        "ClientID": "ABCDEF1234567890ABCDEF10",
        "Timestamp": "2018-07-07T00:20:06",
        "RequestCount": 10500,
        "SampledCount": 31,
        "BorrowCount": 0
    }
]'
{
    "SamplingTargetDocuments": [
        {
            "RuleName": "base-scorekeep",
            "FixedRate": 0.1,
            "ReservoirQuota": 2,
            "ReservoirQuotaTTL": 1530923107.0,
            "Interval": 10
        },
        {
            "RuleName": "polling-scorekeep",
            "FixedRate": 0.003,
            "ReservoirQuota": 0,
            "ReservoirQuotaTTL": 1530923107.0,
            "Interval": 10
        }
    ],
    "LastRuleModification": 1530920505.0,
    "UnprocessedStatistics": []
}
```

La respuesta de X-Ray incluye una cuota que utilizar en lugar de tomarla prestada del depósito. En este ejemplo, el servicio ha tomado prestados 10 rastros desde el depósito en 10 segundos y ha aplicado el porcentaje fijo del 10 por ciento a las otras 100 solicitudes, lo que se traduce en un total de 20 solicitudes muestreadas. La cuota es buena durante cinco minutos (que se indica mediante el tiempo de vida) o hasta que se asigna una nueva cuota. X-Ray también podría asignar un intervalo de informes más largo que el predeterminado, aunque no aquí.

> **Note**
>
> La respuesta de X-Ray podría no incluir una cuota la primera vez que llama. Continue tomando prestado del depósito hasta que se les asigne una cuota.

Los otros dos campos de la respuesta podrían indicar problemas con la entrada. Compruebe `LastRuleModification` respecto a la última vez que llamó a `GetSamplingRules`. Si es más reciente, obtenga una nueva copia de las reglas. `UnprocessedStatistics` puede incluir errores que indican que una regla se ha eliminado, que el documento de estadísticas en la entrada era demasiado antiguo o errores de permisos.

AWS X-RayDocumentos de segmentos de

Un segmento de rastreo es una representación JSON de una solicitud que atiende su aplicación. Un segmento de rastro registra información sobre la solicitud original, información sobre el trabajo que su aplicación realiza localmente y subsegmentos con información sobre las llamadas posteriores que su aplicación realiza a los recursos de AWS, las API HTTP y las bases de datos SQL.

Un documento de segmento transmite información sobre un segmento a X-Ray. Un documento de segmento puede tener un tamaño de hasta 64 kB y contener un segmento completo con subsegmentos, un fragmento de un segmento que indique que una solicitud está en curso o un único subsegmento que se envía por separado. Puede enviar documentos de segmento directamente a X-Ray mediante la API de `PutTraceSegments`.

X-Ray compila y procesa los documentos de segmento para generar resúmenes de rastros y rastros completos que admiten consultas a los que puede obtener acceso mediante las API de `GetTraceSummaries` y `BatchGetTraces`, respectivamente. Además de los segmentos y subsegmentos que envía a X-Ray, el servicio utiliza la información de los subsegmentos para generar segmentos inferidos, que se añaden al rastro completo. Los segmentos inferidos representan servicios y recursos posteriores en el mapa de rastros.

X-Ray proporciona un esquema JSON para los documentos de segmento. Puede descargar el esquema aquí: xray-segmentdocument-schema-v1.0.0. Los campos y objetos incluidos en el esquema se describen con más detalle en las secciones siguientes.

X-Ray indexa un subconjunto de los campos de segmento para su uso con expresiones de filtro. Por ejemplo, si establece el campo `user` de un segmento en un identificador único, puede buscar los segmentos asociados a usuarios específicos en la consola de X-Ray o mediante la API de `GetTraceSummaries`. Para obtener más información, consulte Uso de expresiones de filtro.

Cuando instrumente su aplicación con el SDK de X-Ray, el SDK generará automáticamente los documentos de segmento. En lugar de enviar documentos directamente a X-Ray, el SDK los transmite a través de un puerto UDP local al daemon de X-Ray. Para obtener más información, consulte Envío de documentos de segmento al daemon de X-Ray.

Secciones

Campos de segmentos

Un segmento registra información de rastreo sobre una solicitud que atiende su aplicación. Como mínimo, un segmento registra el nombre, ID, hora de inicio, ID de rastro y el tiempo total de la solicitud.

Example Segmento completo mínimo

```
{
  "name" : "example.com",
  "id" : "70de5b6f19ff9a0a",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  "end_time" : 1.478293361449E9
```

```
}
```

Los siguientes campos son obligatorios o necesarios en determinadas circunstancias para los segmentos.

> **Note**
>
> Los valores deben ser cadenas (de hasta 250 caracteres), a menos que se indique lo contrario.

Campos de segmentos obligatorios

- `name`: nombre lógico del servicio que gestiona la solicitud (hasta 200 caracteres). Por ejemplo, el nombre de su aplicación o el nombre de dominio. Los nombres pueden contener letras Unicode, números y espacios en blanco, así como los siguientes símbolos: `_`, `.`, `:`, `/`, `%`, `&`, `#`, `=`, `+`, `\`, `-`, `@`
- `id`: un identificador de 64 bits para el segmento, único entre otros segmentos del mismo rastro, en 16 dígitos hexadecimales.
- `trace_id`: un identificador único que conecta todos los segmentos y subsegmentos procedentes de una única solicitud de cliente.

  Formato de ID de rastro de X-Ray

  Un `trace_id` de X-Ray consta de tres números separados por guiones. Por ejemplo, `1-58406520-a006649127e371903a2de979`. Esto incluye:

  - El número de versión, que es `1`.
  - La hora en que se realizó la solicitud original, en formato de tiempo Unix, en 8 dígitos hexadecimales.

    Por ejemplo, las 10:00 del 1 de diciembre de 2016, PST en formato de tiempo Unix es `1480615200` segundos o `58406520` en formato hexadecimal.
  - Un identificador 96 bits del rastro, único a nivel global, en 24 dígitos hexadecimales.

  > **Note**
  >
  > Ahora X-Ray admite ID de rastro creados mediante OpenTelemetry o cualquier otro marco que se ajuste a la especificación Trace Context de W3C. Un ID de rastro conforme a la especificación de W3C debe estar formateado en el formato

de ID de rastro de X-Ray cuando se envía a X-Ray. Por ejemplo, el ID de rastro `4efaaf4d1e8720b39541901950019ee5` conforme a la especificación de W3C debe tener el formato `1-4efaaf4d-1e8720b39541901950019ee5` cuando se envíe a X-Ray. Si bien los ID de rastro de X-Ray incluyen la marca de tiempo de la solicitud original en formato de tiempo Unix, no es obligatorio cuando se envían ID de rastro conforme a la especificación de W3C en el formato de X-Ray.

> **Seguridad de ID de rastro**
>
> Los ID de rastro se pueden ver en los encabezados de respuesta. Genere ID de rastro con algoritmo aleatorio seguro para garantizar que los atacantes no puedan calcular futuros ID de rastro y enviar solicitudes con esos ID a su aplicación.

- `start_time`: número que representa el momento en que se creó el segmento, en segundos de punto flotante en formato de tiempo Unix. Por ejemplo, `1480615200.010` o `1.480615200010E9`. Use tantos decimales como necesite. Se recomienda una precisión de microsegundos cuando sea posible.
- `end_time`: número que representa el momento en que se cerró el segmento. Por ejemplo, `1480615200.090` o `1.480615200090E9`. Especifique un `end_time` o `in_progress`.
- `in_progress`: booleano que se establece en `true` en lugar de especificar un `end_time` para registrar que se inició un subsegmento pero que no se completó. Envíe un segmento en curso cuando su aplicación reciba una solicitud que llevará tiempo atender, para rastrear la recepción de la solicitud. Cuando la respuesta se envía, envíe el segmento completo para sobrescribir el segmento en curso. Envíe solo un segmento completo o uno o cero segmentos en curso por solicitud.

> **Nombres de servicio**
>
> El `name` de un segmento debe coincidir con el nombre de dominio o el nombre lógico del servicio que genere el segmento. Sin embargo, este requisito no se exige. Cualquier aplicación que tenga permiso para `PutTraceSegments` puede enviar segmentos con cualquier nombre.

Los siguientes campos son opcionales para los segmentos.

Campos de segmentos opcionales

- `service`: un objeto con información sobre su aplicación.
  - `version`: una cadena que identifica la versión de la aplicación que atiende la solicitud.
- `user`: una cadena que identifica el usuario que envía la solicitud.
- `origin`: el tipo de recurso de AWS que ejecuta la aplicación.

  Valores admitidos

  - `AWS::EC2::Instance`: una instancia de Amazon EC2.
  - `AWS::ECS::Container`: un contenedor de Amazon ECS.
  - `AWS::ElasticBeanstalk::Environment`: un entorno de Elastic Beanstalk

  Si hay varios valores que sean aplicables a su aplicación, utilice el que sea más específico. Por ejemplo, supongamos que un Docker multicontenedor en Elastic Beanstalk ejecuta su aplicación en un contenedor de Amazon ECS que, a su vez, se ejecuta en una instancia de Amazon EC2. En este caso, el origen se establecería en `AWS::ElasticBeanstalk::Environment`, puesto que es el elemento principal de los otros dos recursos.
- `parent_id`: un ID de subsegmento que especifica si la solicitud se originó desde una aplicación instrumentada. El SDK de X-Ray añade el ID del subsegmento principal al encabezado de rastreo para las llamadas HTTP posteriores. En el caso de subsegmentos anidados, un subsegmento puede tener un segmento o un subsegmento como padre.
- `http`: objetos `http` con información sobre la solicitud HTTP original.
- `aws`: objeto `aws` con información sobre el recurso de AWS en el que la aplicación atiende la solicitud.
- `error`, `throttle`, `fault` y `cause`: campos de error que indican que se ha producido un error y que incluyen información sobre la excepción que ha causado el error.
- `annotations` – `annotations` objeto con pares de clave-valor que X-Ray debe indexar para las búsquedas.
- `metadata` – `metadata` objeto con los datos adicionales que desea almacenar en el segmento.
- `subsegments`: matriz de objetos `subsegment`.

Subsegmentos

Puede crear subsegmentos para registrar las llamadas a los recursos y Servicios de AWS que ha realizado con el SDK de AWS, las llamadas a las API web HTTP internas o externas, o las consultas

de base de datos SQL. También puede crear subsegmentos para depurar o anotar bloques de código en su aplicación. Los subsegmentos pueden contener otros subsegmentos, por lo que un subsegmento personalizado que registre los metadatos de una llamada a una función interna puede contener otros subsegmentos personalizados y subsegmentos para llamadas posteriores.

Un subsegmento registra una llamada posterior desde el punto de vista del servicio que llama. X-Ray utiliza subsegmentos para identificar los servicios posteriores que no envían segmentos y crean entradas para segmentos en el gráfico de servicios.

Un subsegmento puede incrustarse en un documento de segmentos completos o enviarse por separado. Envíe subsegmentos por separado para realizar un rastreo asíncrono de las llamadas posteriores para realizar solicitudes de larga ejecución o para evitar superar el tamaño máximo del documento de segmentos.

Example Segmento con un subsegmento incrustado

Un subsegmento independiente tiene un `type` de `subsegment` y un `parent_id` que identifica el segmento principal.

```
{
  "trace_id"   : "1-5759e988-bd862e3fe1be46a994272793",
  "id"         : "defdfd9912dc5a56",
  "start_time" : 1461096053.37518,
  "end_time"   : 1461096053.4042,
  "name"       : "www.example.com",
  "http"       : {
    "request"  : {
      "url"        : "https://www.example.com/health",
      "method"     : "GET",
      "user_agent" : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)
 AppleWebKit/601.7.7",
      "client_ip"  : "11.0.3.111"
    },
    "response" : {
      "status"         : 200,
      "content_length" : 86
    }
  },
  "subsegments" : [
    {
      "id"         : "53995c3f42cd8ad8",
      "name"       : "api.example.com",
```

```
      "start_time" : 1461096053.37769,
      "end_time"   : 1461096053.40379,
      "namespace"  : "remote",
      "http"       : {
        "request"  : {
          "url"    : "https://api.example.com/health",
          "method" : "POST",
          "traced" : true
        },
        "response" : {
          "status"         : 200,
          "content_length" : 861
        }
      }
    }
  ]
}
```

En el caso de las solicitudes de larga ejecución, puede enviar un segmento en curso para notificar a X-Ray que la solicitud se ha recibido y, a continuación, enviar por separado los subsegmentos para que se rastreen antes de que se complete la solicitud original.

Example Segmento en curso

```
{
  "name" : "example.com",
  "id" : "70de5b6f19ff9a0b",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  "in_progress": true
}
```

Example Subsegmentos independientes

Un subsegmento independiente tiene un `type` de `subsegment`, un `trace_id` y un `parent_id` que identifica el segmento principal.

```
{
  "name" : "api.example.com",
  "id" : "53995c3f42cd8ad8",
  "start_time" : 1.478293361271E9,
  "end_time" : 1.478293361449E9,
  "type" : "subsegment",
```

```
  "trace_id" : "1-581cf771-a006649127e371903a2de979"
  "parent_id" : "defdfd9912dc5a56",
  "namespace"  : "remote",
  "http"       : {
      "request"  : {
          "url"    : "https://api.example.com/health",
          "method" : "POST",
          "traced" : true
      },
      "response" : {
          "status"         : 200,
          "content_length" : 861
      }
  }
}
```

Cuando se complete la solicitud, cierre el segmento reenviándolo con un `end_time`. El segmento completo sobrescribe el segmento en curso.

También puede enviar subsegmentos por separado para solicitudes realizadas que activen flujos de trabajo asíncronos. Por ejemplo, una API web puede devolver una respuesta `OK 200` inmediatamente antes de iniciar el trabajo que el usuario ha solicitado. Puede enviar un segmento completo a X-Ray en cuanto se envíe la respuesta, seguido de subsegmentos para el trabajo realizado más adelante. Al igual que con los segmentos, también puede enviar un fragmento de un subsegmento para registrar que el subsegmento se ha iniciado y, a continuación, sobrescribirlo con un subsegmento completo una vez que se haya completado la llamada posterior.

Los siguientes campos son obligatorios o necesarios en determinadas circunstancias para los subsegmentos.

> **Note**
>
> Los valores son cadenas (de hasta 250 caracteres), a menos que se indique lo contrario.

Campos de subsegmentos obligatorios

- `id`: un identificador de 64 bits para el subsegmento, único entre otros segmentos del mismo rastro, en 16 dígitos hexadecimales.
- `name`: nombre lógico del subsegmento. En el caso de las llamadas posteriores, asigne un nombre al subsegmento detrás del recurso o servicio llamado. Para los subsegmentos personalizados,

asigne el nombre al subsegmento detrás del código que lo instrumenta (por ejemplo, un nombre de función).

- `start_time`: número que representa el momento en que se creó el subsegmento, en segundos de punto flotante en tiempo Unix, con una precisión de milisegundos. Por ejemplo, `1480615200.010` o `1.480615200010E9`.
- `end_time`: número que representa el momento en que se cerró el subsegmento. Por ejemplo, `1480615200.090` o `1.480615200090E9`. Especifique un valor para `end_time` o `in_progress`.
- `in_progress`: booleano que se establece en `true` en lugar de especificar un `end_time` para registrar que se inició un subsegmento pero que no se completó. Envíe solo un subsegmento completo y uno o cero subsegmentos en curso por solicitud posterior.
- `trace_id`: ID de rastro del segmento principal del subsegmento. Solo es necesario si el envío del subsegmento se realiza por separado.

  Formato de ID de rastro de X-Ray

  Un `trace_id` de X-Ray consta de tres números separados por guiones. Por ejemplo, `1-58406520-a006649127e371903a2de979`. Esto incluye:

  - El número de versión, que es 1.
  - La hora en que se realizó la solicitud original, en formato de tiempo Unix, en 8 dígitos hexadecimales.

    Por ejemplo, las 10:00 del 1 de diciembre de 2016, PST en formato de tiempo Unix es `1480615200` segundos o `58406520` en formato hexadecimal.
  - Un identificador 96 bits del rastro, único a nivel global, en 24 dígitos hexadecimales.

  > **Note**
  >
  > Ahora X-Ray admite ID de rastro creados mediante OpenTelemetry o cualquier otro marco que se ajuste a la especificación Trace Context de W3C. Un ID de rastro conforme a la especificación de W3C debe estar formateado en el formato de ID de rastro de X-Ray cuando se envía a X-Ray. Por ejemplo, el ID de rastro `4efaaf4d1e8720b39541901950019ee5` conforme a la especificación de W3C debe tener el formato `1-4efaaf4d-1e8720b39541901950019ee5` cuando se envíe a X-Ray. Si bien los ID de rastro de X-Ray incluyen la marca de tiempo de la solicitud original en

formato de tiempo Unix, no es obligatorio cuando se envían ID de rastro conforme a la especificación de W3C en el formato de X-Ray.

- `parent_id`: ID del segmento principal del subsegmento. Solo es necesario si el envío del subsegmento se realiza por separado. En el caso de subsegmentos anidados, un subsegmento puede tener un segmento o un subsegmento como padre.
- `type` – `subsegment`. Solo es necesario si el envío del subsegmento se realiza por separado.

Los siguientes campos son opcionales para los subsegmentos.

Campos de subsegmentos opcionales

- `namespace`: `aws` para las llamadas al SDK de AWS; `remote` para las demás llamadas posteriores.
- `http`: objeto `http` con información sobre una llamada HTTP saliente.
- `aws`: objeto `aws` con información sobre el recurso de AWS posterior al que ha llamado la aplicación.
- `error`, `throttle`, `fault` y `cause`: campos de error que indican que se ha producido un error y que incluyen información sobre la excepción que ha causado el error.
- `annotations`: objeto `annotations` con pares de clave-valor que X-Ray debe indexar para las búsquedas.
- `metadata`: objeto `metadata` con los datos adicionales que desea almacenar en el segmento.
- `subsegments`: matriz de objetos `subsegment`.
- `precursor_ids`: matriz de identificadores de subsegmento que identifica los subsegmentos con el mismo segmento principal que se completó antes de este subsegmento.

Datos de solicitudes HTTP

Utilice un bloque HTTP para registrar los detalles de una solicitud HTTP que ha atendido su aplicación (en un segmento) o que ha realizado la aplicación en una API HTTP posterior (en un subsegmento). La mayoría de los campos de este objeto se asignan a la información proporcionada en una solicitud y respuesta HTTP.

`http`

Todos los campos son opcionales.

- `request`: información sobre una solicitud.
  - `method`: el método de solicitud. Por ejemplo, `GET`.
  - `url`: la dirección URL completa, obtenida del protocolo, nombre de host y ruta de la solicitud.
  - `user_agent`: la cadena de agente de usuario del cliente del solicitante.
  - `client_ip`: la dirección IP del solicitante. Se puede recuperar del `Source Address` del paquete de direcciones IP o, en el caso de las solicitudes reenviadas, de un encabezado `X-Forwarded-For`.
  - `x_forwarded_for`: (solo segmentos) booleano que indica que se ha leído el `client_ip` desde un encabezado `X-Forwarded-For` y no es fiable, ya que podría haberse falsificado.
  - `traced`: (solo subsegmentos) booleano que indica que la llamada posterior es a otro servicio rastreado. Si este campo está establecido en `true`, X-Ray considera que el rastro se debe dividir hasta que el servicio posterior cargue el segmento con un `parent_id` que coincida con el `id` del subsegmento que contiene este bloque.
- `response`: información sobre una respuesta.
  - `status`: entero que indica el estado HTTP de la respuesta.
  - `content_length`: entero que indica la longitud del cuerpo de la respuesta en bytes.

Cuando instrumente una llamada a una API web posterior, este campo registra un subsegmento con información sobre la solicitud HTTP y la respuesta. X-Ray utiliza el subsegmento para generar un segmento inferido de la API remota.

Example Segmento para las llamadas HTTP atendidas por una aplicación en ejecución en Amazon EC2

```
{
  "id": "6b55dcc497934f1a",
  "start_time": 1484789387.126,
  "end_time": 1484789387.535,
  "trace_id": "1-5880168b-fd5158284b67678a3bb5a78c",
  "name": "www.example.com",
  "origin": "AWS::EC2::Instance",
  "aws": {
    "ec2": {
      "availability_zone": "us-west-2c",
      "instance_id": "i-0b5a4678fc325bg98"
    },
    "xray": {
```

```
        "sdk_version": "2.11.0 for Java"
    },
  },
  "http": {
    "request": {
      "method": "POST",
      "client_ip": "78.255.233.48",
      "url": "http://www.example.com/api/user",
      "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101
 Firefox/45.0",
      "x_forwarded_for": true
    },
    "response": {
      "status": 200
    }
  }
```

Example Subsegmento para una llamada HTTP posterior

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example Segmento inferido para una llamada HTTP posterior

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
```

```
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

Anotaciones

Los segmentos y subsegmentos pueden incluir un objeto `annotations` con uno o varios campos que X-Ray indexa para su uso con expresiones de filtro. Los campos pueden tener valores de cadena, numéricos o booleanos (pero no objetos ni matrices). X-Ray indexa hasta 50 anotaciones por rastro.

Example Segmento para llamadas HTTP con anotaciones

```
{
  "id": "6b55dcc497932f1a",
  "start_time": 1484789187.126,
  "end_time": 1484789187.535,
  "trace_id": "1-5880168b-fd515828bs07678a3bb5a78c",
  "name": "www.example.com",
  "origin": "AWS::EC2::Instance",
  "aws": {
    "ec2": {
      "availability_zone": "us-west-2c",
      "instance_id": "i-0b5a4678fc325bg98"
    },
    "xray": {
        "sdk_version": "2.11.0 for Java"
    },
  },
  "annotations": {
    "customer_category" : 124,
```

```
    "zip_code" : 98101,
    "country" : "United States",
    "internal" : false
  },
  "http": {
    "request": {
      "method": "POST",
      "client_ip": "78.255.233.48",
      "url": "http://www.example.com/api/user",
      "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101
 Firefox/45.0",
      "x_forwarded_for": true
    },
    "response": {
      "status": 200
    }
  }
```

Para que las claves funcionen con filtros, deben estar en orden alfanumérico. Se admite el guion bajo, pero no otros símbolos ni espacios en blanco.

Metadatos

Los segmentos y subsegmentos pueden incluir un objeto `metadata` que contenga uno o varios campos con valores de cualquier tipo, incluidos objetos y matrices. X-Ray no indexa los metadatos, y los valores pueden ser de cualquier tamaño siempre que el documento del segmento no supere el tamaño máximo (64 kB). Puede ver los metadatos en el documento del segmento completo devuelto por la API de `BatchGetTraces`. Las claves de campo (`debug` en el siguiente ejemplo) empezando por `AWS.` están reservadas para su uso por los SDK y los clientes proporcionados por AWS.

Example Subsegmento personalizado con metadatos

```
{
  "id": "0e58d2918e9038e8",
  "start_time": 1484789387.502,
  "end_time": 1484789387.534,
  "name": "## UserModel.saveUser",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

```
  "subsegments": [
    {
      "id": "0f910026178b71eb",
      "start_time": 1484789387.502,
      "end_time": 1484789387.534,
      "name": "DynamoDB",
      "namespace": "aws",
      "http": {
        "response": {
          "content_length": 58,
          "status": 200
        }
      },
      "aws": {
        "table_name": "scorekeep-user",
        "operation": "UpdateItem",
        "request_id": "3AIENM5J4ELQ3SPODHKBIRVIC3VV4KQNSO5AEMVJF66Q9ASUAAJG",
        "resource_names": [
          "scorekeep-user"
        ]
      }
    }
  ]
}
```

Datos de recursos de AWS

Para los segmentos, el objeto `aws` contiene información sobre el recurso en el que se ejecuta la aplicación. Se pueden aplicar varios campos a un solo recurso. Por ejemplo, una aplicación que se ejecute en un entorno Docker multicontenedor en Elastic Beanstalk podría tener información sobre la instancia de Amazon EC2, el contenedor de Amazon ECS que se ejecuta en la instancia y el propio entorno de Elastic Beanstalk.

`aws` (Segmentos)

Todos los campos son opcionales.

- `account_id`: si la aplicación envía segmentos a una Cuenta de AWS diferente, este campo registra el ID de la cuenta que ejecuta la aplicación.
- `cloudwatch_logs`: matriz de objetos que describen un único grupo de registro de CloudWatch.
  - `log_group`: el nombre del grupo de registro de CloudWatch.

  - `arn`— El ARN del grupo de registro de CloudWatch.
- `ec2`: información sobre una instancia de Amazon EC2.
  - `instance_id`: el ID de la instancia de EC2.
  - `instance_size`: el tipo de la instancia de EC2.
  - `ami_id`: el ID de imagen de máquina de Amazon.
  - `availability_zone`: la zona de disponibilidad en la que se ejecuta la instancia.
- `ecs`: información sobre un contenedor de Amazon ECS.
  - `container`: el nombre de host de su contenedor.
  - `container_id`: el ID completo de su contenedor.
  - `container_arn`: el ARN de su instancia de contenedor.
- `eks`: la información acerca de un clúster de Amazon EKS.
  - `pod`: el nombre de host de su pod de EKS.
  - `cluster_name`: el nombre del clúster de EKS.
  - `container_id`: el ID completo de su contenedor.
- `elastic_beanstalk`: información sobre un entorno de Elastic Beanstalk. Puede encontrar esta información en un archivo denominado `/var/elasticbeanstalk/xray/environment.conf` en las plataformas más recientes de Elastic Beanstalk.
  - `environment_name`: el nombre del entorno.
  - `version_label`: el nombre de la versión de la aplicación que está implementada actualmente en la instancia que ha atendido la solicitud.
  - `deployment_id`: número que indica el ID de la última implementación satisfactoria en la instancia que ha atendido la solicitud.
- `xray`: metadatos sobre el tipo y la versión de la instrumentación utilizada.
  - `auto_instrumentation`: booleano que indica si se utilizó la instrumentación automática (por ejemplo, el agente Java).
  - `sdk_version`. La versión del SDK o del agente que se está utilizando.
  - `sdk`: el tipo de SDK.

Example Bloque de AWS con complementos

```
"aws":{
   "elastic_beanstalk":{
```

```
      "version_label":"app-5a56-170119_190650-stage-170119_190650",
      "deployment_id":32,
      "environment_name":"scorekeep"
   },
   "ec2":{
      "availability_zone":"us-west-2c",
      "instance_id":"i-075ad396f12bc325a",
      "ami_id":
   },
   "cloudwatch_logs":[
      {
         "log_group":"my-cw-log-group",
         "arn":"arn:aws:logs:us-west-2:012345678912:log-group:my-cw-log-group"
      }
   ],
   "xray":{
      "auto_instrumentation":false,
      "sdk":"X-Ray for Java",
      "sdk_version":"2.8.0"
   }
}
```

Para los subsegmentos, este campo registra información acerca de los recursos y Servicios de AWS a los que obtiene acceso la aplicación. X-Ray utiliza esta información para crear segmentos inferidos que representan los servicios posteriores del mapa de servicio.

`aws` (subsegmentos)

Todos los campos son opcionales.

- `operation`: el nombre de la acción de la API invocada para un recurso o Servicio de AWS.
- `account_id`: si la aplicación obtiene acceso a recursos en una cuenta diferente o envía segmentos a una cuenta diferente, este campo registra el ID de la cuenta propietaria del recurso de AWS al que obtiene acceso la aplicación.
- `region`: si el recurso está en una región diferente a la de la aplicación, este campo registra la región. Por ejemplo, `us-west-2`.
- `request_id`: un identificador único para la solicitud.
- `queue_url`: para las operaciones incluidas en una cola de Amazon SQS, la dirección URL de la cola.
- `table_name`: para las operaciones de una tabla de DynamoDB, el nombre de la tabla.

Example Subsegmento para una llamada a DynamoDB con el fin de guardar un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Errores y excepciones

Cuando se produzca un error, puede registrar información sobre el error y las excepciones que se generan. Registre los errores en los segmentos cuando la aplicación devuelva un error al usuario, y en los subsegmentos cuando una llamada posterior devuelva un error.

tipos de error

Establezca uno o varios de los campos siguientes en `true` para indicar que se ha producido un error. Se pueden aplicar varios tipos si se trata de errores compuestos. Por ejemplo, un error `429 Too Many Requests` de una llamada posterior podría hacer que la aplicación devolviera un error `500 Internal Server Error`, en cuyo caso se aplicarían los tres tipos.

- `error`: booleano que indica que se ha producido un error del cliente (el código de estado de la respuesta fue 4XX Client Error).
- `throttle`: booleano que indica que se ha limitado una solicitud (el código de estado de la respuesta fue 429 Too Many Requests).
- `fault`: booleano que indica que se ha producido un error del servidor (el código de estado de la respuesta fue 5XX Server Error).

Indique la causa del error incluyendo un objeto cause en el segmento o subsegmento.

`cause`

Una causa puede ser un ID de excepción de 16 caracteres o un objeto con los siguientes campos:

- `working_directory`: la ruta completa del directorio de trabajo donde se produjo la excepción.
- `paths`: la matriz de rutas a las bibliotecas o módulos que se estaban usando cuando se produjo la excepción.
- `exceptions`: la matriz de objetos exception.

Incluya información detallada sobre el error en uno o varios objetos exception.

`exception`

Todos los campos son opcionales.

- `id`: un identificador de 64 bits para la excepción, único entre otros segmentos del mismo rastro, en 16 dígitos hexadecimales.
- `message`: el mensaje de excepción.
- `type`: el tipo de excepción.
- `remote`: booleano que indica que la excepción se produjo por un error devuelto por un servicio posterior.
- `truncated`: entero que indica el número de marcos de pila que se han omitido de `stack`.
- `skipped`: entero que indica el número de excepciones que se omitieron entre esta excepción y su excepción secundaria, es decir, la excepción que la ha causado.
- `cause`: ID de excepción del elemento principal de la excepción, es decir, la excepción que provocó esta excepción.
- `stack`: matriz de objetos stackFrame.

Si está disponible, registre la información acerca de la pila de llamada en objetos stackFrame.

`stackFrame`

Todos los campos son opcionales.

- `path`: la ruta relativa al archivo.

- `line`: la línea dentro del archivo.
- `label`: el nombre de la función o método.

Consultas SQL

Puede crear subsegmentos para las consultas que la aplicación realiza en una base de datos SQL.

`sql`

Todos los campos son opcionales.

- `connection_string`: para SQL Server u otras conexiones de base de datos que no usen cadenas de conexión URL, este campo registra la cadena de conexión, sin incluir las contraseñas.
- `url`: para una conexión a la base de datos que utilice una cadena de conexión URL, este campo registra la dirección URL, sin incluir las contraseñas.
- `sanitized_query`: la consulta a la base de datos, con todos los valores proporcionados por el usuario eliminados o reemplazados con un marcador de posición.
- `database_type`: el nombre del motor de base de datos.
- `database_version`: el número de versión del motor de base de datos.
- `driver_version`: el nombre y número de versión del controlador del motor de base de datos que usa la aplicación.
- `user`: el nombre de usuario de la base de datos.
- `preparation`: `call` si la consulta utiliza un `PreparedCall`; `statement` si la consulta utiliza un `PreparedStatement`.

Example Subsegmento con una consulta SQL

```
{
  "id": "3fd8634e78ca9560",
  "start_time": 1484872218.696,
  "end_time": 1484872218.697,
  "name": "ebdb@aawijb5u25wdoy.cpamxznpdoq8.us-west-2.rds.amazonaws.com",
  "namespace": "remote",
  "sql" : {
    "url": "jdbc:postgresql://aawijb5u25wdoy.cpamxznpdoq8.us-
west-2.rds.amazonaws.com:5432/ebdb",
    "preparation": "statement",
```

```
    "database_type": "PostgreSQL",
    "database_version": "9.5.4",
    "driver_version": "PostgreSQL 9.4.1211.jre7",
    "user" : "dbuser",
    "sanitized_query" : "SELECT  *  FROM  customers  WHERE  customer_id=?;"
  }
}
```

AWS X-RayConceptos de

Los servicios envían datos a AWS X-Ray como segmentos. A continuación, X-Ray agrupa los segmentos que tienen una solicitud en común en rastros. X-Ray procesa los rastros para generar un gráfico de servicios, que constituye una representación visual de la aplicación.

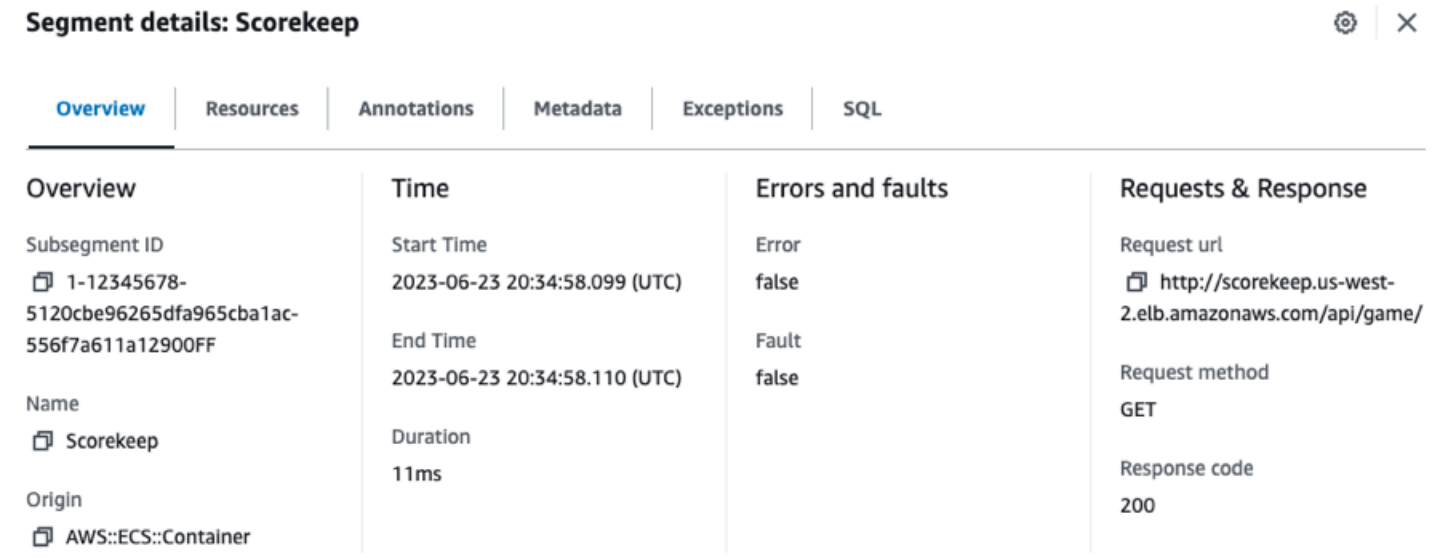
Conceptos

- Segmentos
- Subsegmentos
- Gráfico de servicios
- Registros de seguimiento
- Muestreo
- Encabezado de seguimiento
- Expresiones de filtro
- Grupos
- Anotaciones y metadatos
- Errores y excepciones

Segmentos

Los recursos informáticos en los que se ejecuta la lógica de su aplicación envían datos del trabajo en forma de segmentos. Un segmento incluye el nombre del recurso, detalles de la solicitud y detalles del trabajo realizado. Por ejemplo, cuando una solicitud HTTP llega a la aplicación, puede registrar datos sobre:

- El host nombre de host, alias o dirección IP
- La solicitud: método, dirección del cliente, ruta y agente de usuario
- La respuesta: estado y contenido
- El trabajo realizado: hora de inicio y finalización y subsegmentos
- Problemas que se producen: errores, fallos y excepciones, incluida la captura automática de pilas de excepciones.



El SDK de X-Ray recopila información a partir de los encabezados de solicitud y respuesta, el código en su aplicación y de los metadatos sobre los recursos de AWS donde se ejecuta el kit. Puede elegir los datos que se recopilan. Para hacerlo, debe modificar la configuración o el código de la aplicación para instrumentar las solicitudes entrantes, las solicitudes posteriores y los clientes del SDK de AWS.

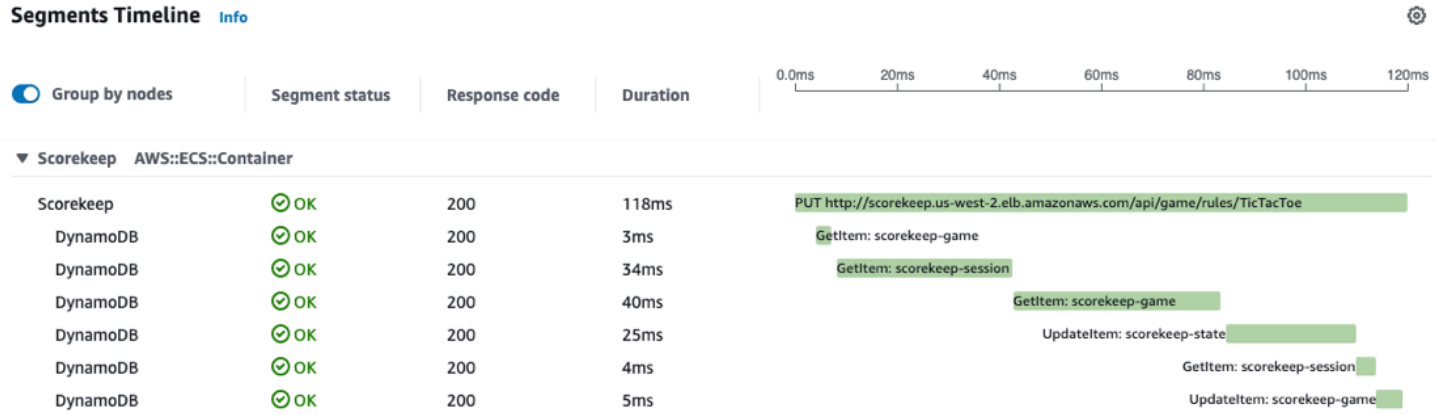
> **Solicitudes reenviadas**
>
> Si un equilibrador de carga u otro intermediario reenvía una solicitud a la aplicación, X-Ray toma la IP de cliente del encabezado `X-Forwarded-For` de la solicitud en lugar de tomar la IP de origen del paquete IP. La IP de cliente que se graba para una solicitud reenviada puede estar falsificada, por lo que no se debe confiar en ella.

Puede utilizar el SDK de X-Ray para registrar información adicional como, por ejemplo, anotaciones y metadatos. Para obtener más información sobre la estructura y la información que se registra en los segmentos y subsegmentos, consulte AWS X-RayDocumentos de segmentos de. Los documentos de segmento pueden tener un tamaño de hasta 64 kB.

Subsegmentos

Un segmento puede desglosar los datos del trabajo realizado en subsegmentos. Los subsegments proporcionan información y detalles más precisos sobre los intervalos de las llamadas posteriores que la aplicación realizó para cumplir la solicitud original. Un subsegmento puede contener detalles adicionales sobre una llamada a un Servicio de AWS, una API HTTP externa, o una base de datos

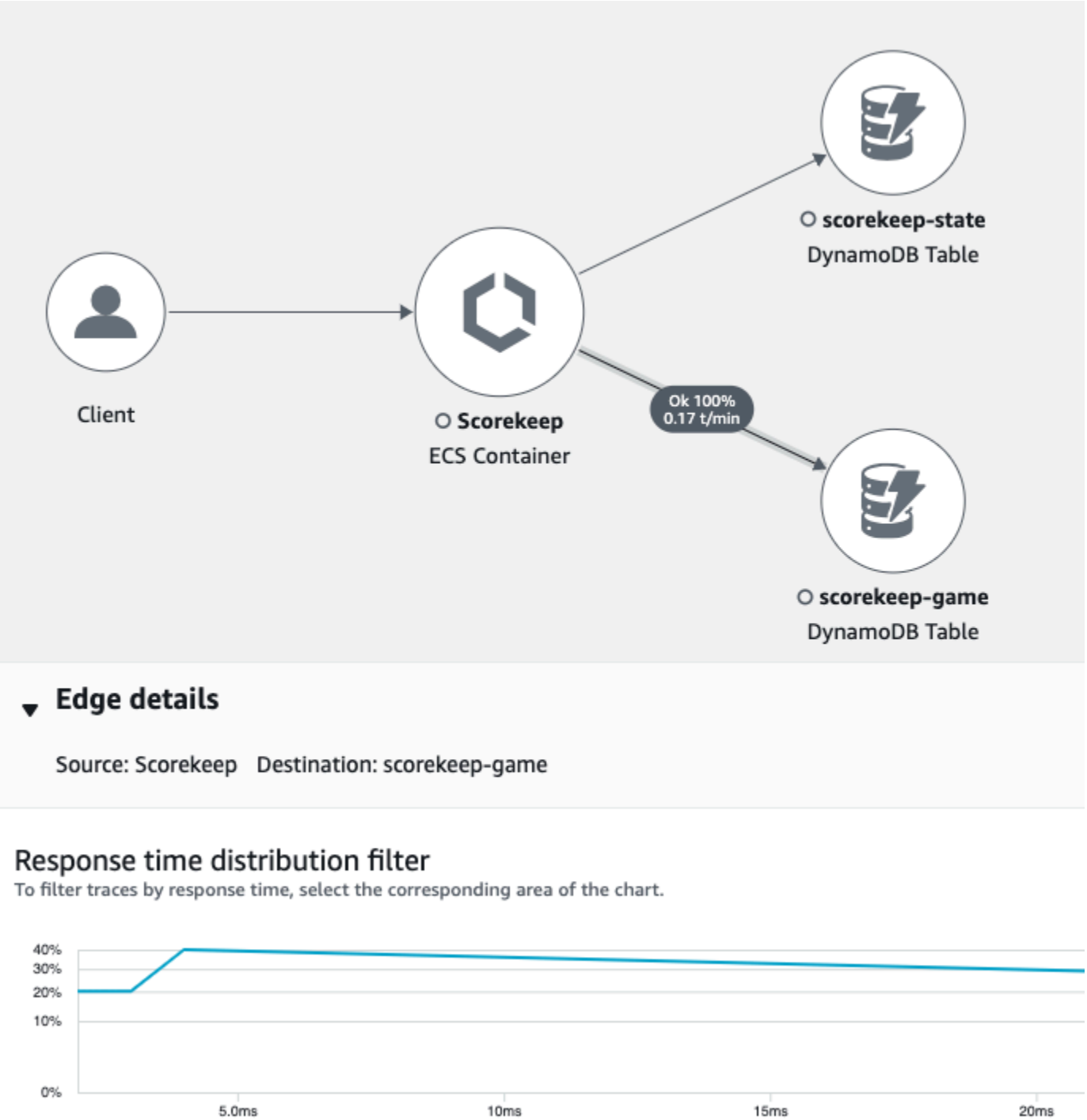
SQL. Puede incluso definir subsegmentos arbitrarios para instrumentar funciones específicas o líneas de código en su aplicación.



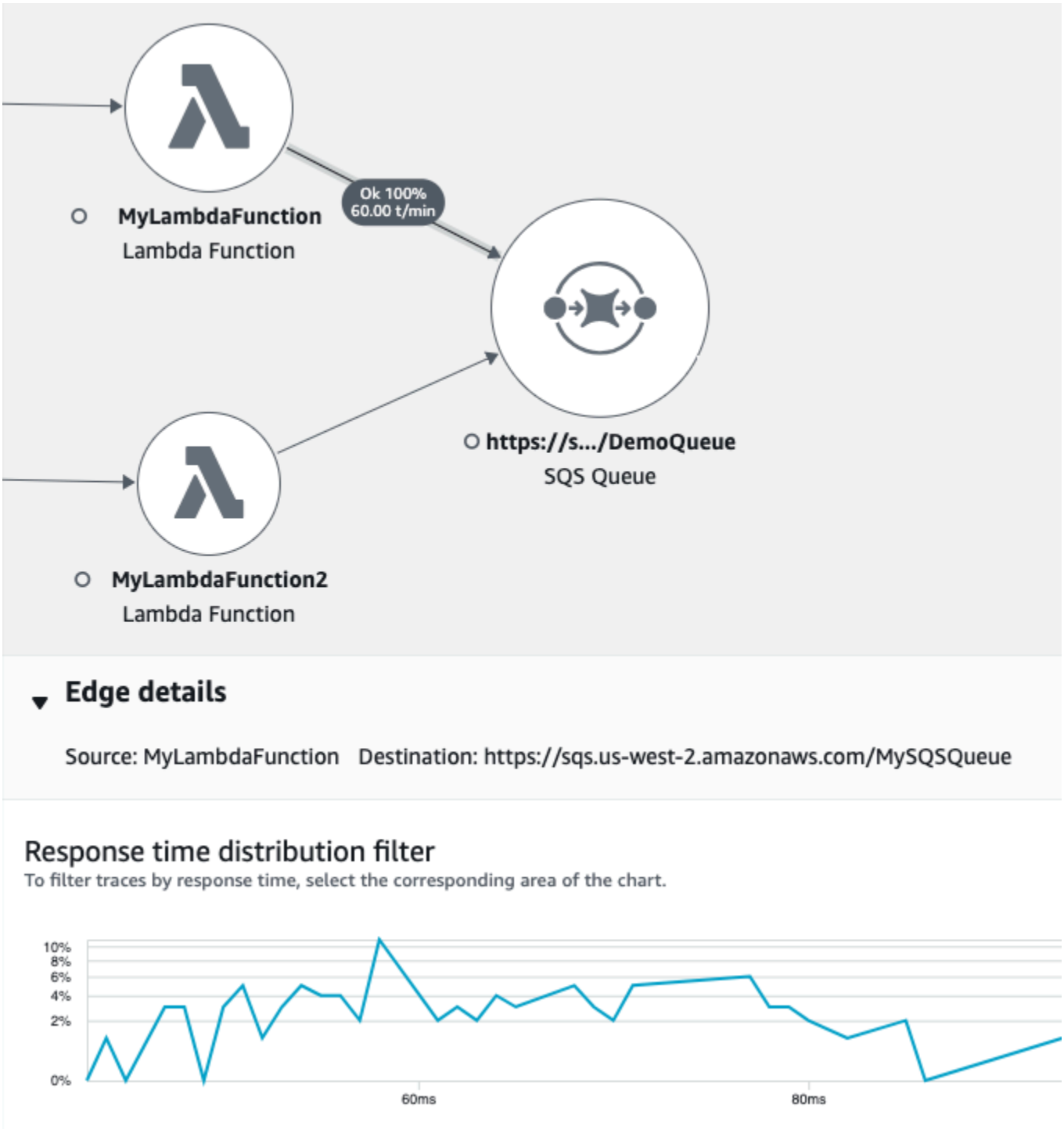
En el caso de los servicios que no envían sus propios segmentos, como Amazon DynamoDB, X-Ray utiliza subsegmentos para generar segmentos inferidos y nodos posteriores en el mapa de rastros. Esto le permite ver todas sus dependencias posteriores, incluso si no soportan el rastreo o son externas.

Los subsegmentos representan la vista de su aplicación de una llamada posterior como cliente. Si el servicio posterior también está instrumentado, el segmento que envía sustituye al segmento inferido generado desde el subsegmento del cliente principal. El nodo en el gráfico de servicio utiliza siempre información desde el segmento del servicio, si está disponible, mientras que el límite entre los dos nodos utiliza el subsegmento del servicio ascendente.

Por ejemplo, si llama a DynamoDB con un cliente instrumentado del SDK de AWS, el SDK de X-Ray registra un subsegmento para dicha llamada. DynamoDB no envía un segmento, por lo que el segmento inferido en el rastro, el nodo de DynamoDB en el gráfico de servicio y el borde entre su servicio y DynamoDB contienen información del subsegmento.



Si llama a otro servicio instrumentado con una aplicación instrumentada, el servicio posterior envía su propio segmento para registrar su vista de la misma llamada que el servicio ascendente registró en un subsegmento. En el gráfico de servicio, los nodos de ambos servicios contienen información de tiempo y error de los segmentos de dichos servicios, mientras que el límite entre ellos contiene información del subsegmento del servicio ascendente.



Ambos puntos de vista son útiles, ya que el servicio posterior registra precisamente cuándo comenzó y finalizó el trabajo en la solicitud y el servicio ascendente registra la latencia de ida y vuelta, incluido el tiempo que la solicitud ha pasado viajando entre los dos servicios.

Gráfico de servicios

X-Ray utiliza los datos que su aplicación envía para generar un gráfico de servicios. Cada recurso de AWS que envía datos a X-Ray aparece como un servicio en el gráfico. Los límites conectan los servicios que trabajan en equipo para atender solicitudes. Los límites conectan a los clientes con la aplicación, y conectan la aplicación con los servicios y recursos posteriores que esta utiliza.

> **Nombres de servicio**
>
> El `name` de un segmento debe coincidir con el nombre de dominio o el nombre lógico del servicio que genere el segmento. Sin embargo, este requisito no se exige. Cualquier aplicación que tenga permiso para `PutTraceSegments` puede enviar segmentos con cualquier nombre.

Un gráfico de servicios es un documento JSON que contiene información acerca de los servicios y recursos que componen la aplicación. La consola de X-Ray utiliza el gráfico de servicios para generar una visualización o un mapa de servicio.

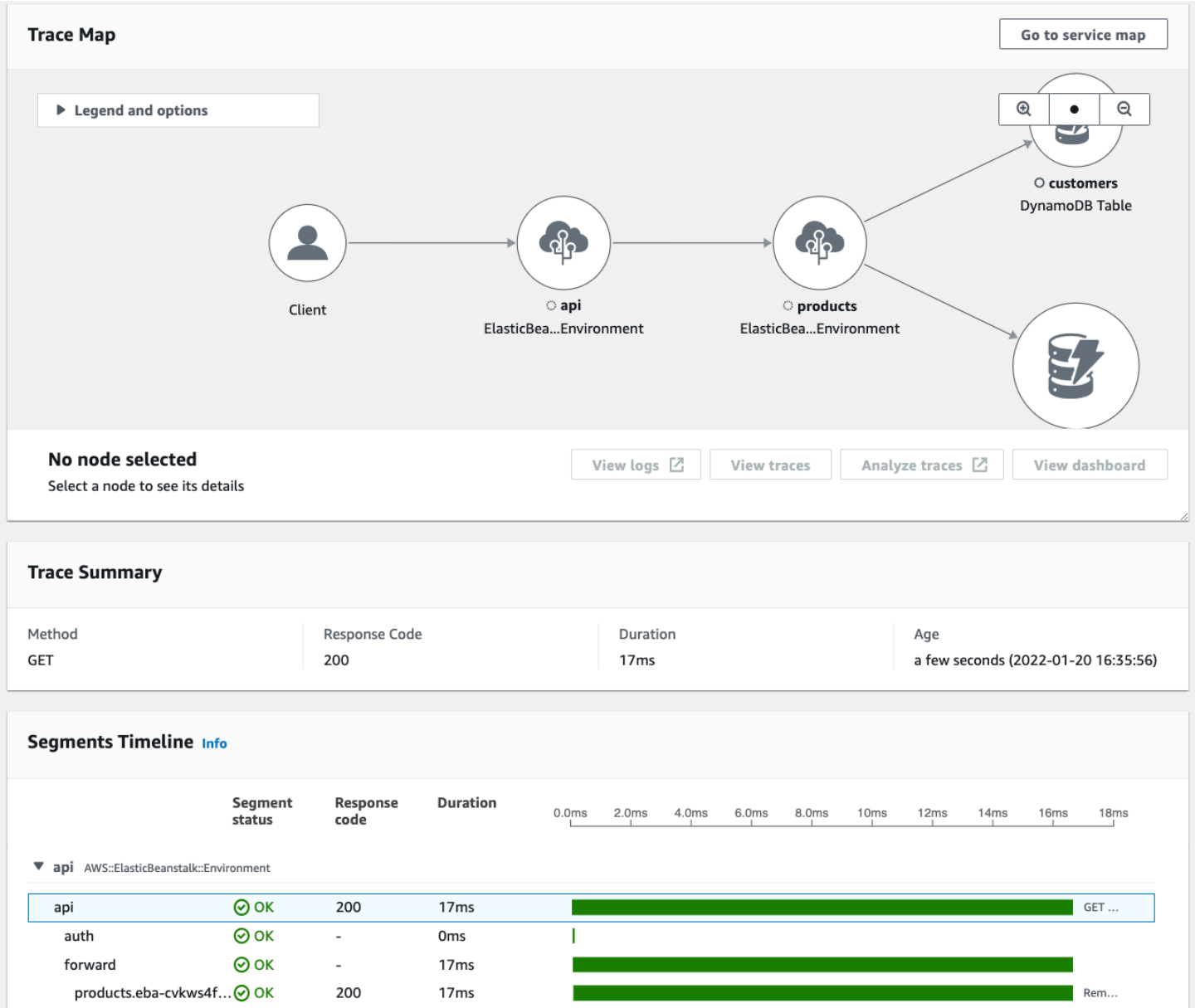
En las aplicaciones distribuidas, X-Ray combina nodos de todos los servicios que procesan solicitudes con el mismo ID de rastro en un único gráfico de servicios. El primer servicio al que llega la solicitud añade un encabezado de rastro que se propaga entre el front-end y servicios que a los llama.

Por ejemplo, Scorekeep ejecuta una API web que llama a un microservicio (una función de AWS Lambda) para generar un nombre aleatorio usando una biblioteca Node.js. El SDK de X-Ray para Java genera el ID de rastro y lo incluye en las llamadas a Lambda. Lambda envía datos de rastreo y transfiere el ID de rastro a la función. El SDK de X-Ray para Node.js también utiliza el ID de rastro para enviar datos. En consecuencia, el mapa de rastros muestra los nodos de la API, el servicio de Lambda y la función de Lambda como nodos independientes pero conectados.

Los datos del gráfico de servicio se conservan durante 30 días.

Registros de seguimiento

El ID de rastro muestra la ruta que recorre una solicitud en la aplicación. Un rastro recopila todos los segmentos que genera una solicitud. La solicitud suele ser una solicitud GET o una solicitud POST de HTTP que viaja a través de un equilibrador de carga, llega a su código de aplicación y genera llamadas posteriores a otros servicios de AWS o API web externas. El primer servicio compatible con el que interactúa la solicitud HTTP le añade un encabezado de ID de rastro, que continúa propagándose para registrar la latencia, la disposición y otros datos que requiere la solicitud.



Consulte Precios de AWS X-Ray para obtener información sobre cómo se facturan los rastros de X-Ray. Los datos de rastro se conservan durante 30 días.

Muestreo

Para garantizar un rastreo eficaz y proporcionar una muestra representativa de las solicitudes que su aplicación atiende, el SDK de X-Ray aplica un algoritmo de muestreo para determinar qué solicitudes se rastrearán. De forma predeterminada, el SDK de X-Ray registra la primera solicitud cada segundo y el 5 % de las solicitudes adicionales.

Para evitar incurrir en cargos por servicio al empezar, la tasa de muestreo predeterminada es conservadora. Puede configurar X-Ray para modificar la regla de muestreo predeterminada y configurar reglas adicionales que aplican el muestreo en función de las propiedades del servicio o solicitud.

Por ejemplo, es posible que desee deshabilitar las solicitudes de muestreo y rastreo completo para las llamadas que modifican el estado o tratar con usuarios o transacciones. Para llamadas de solo lectura de alto volumen, como sondeo en segundo plano, comprobaciones de estado o mantenimiento de conexión, puede muestrear con un bajo índice y seguirá obteniendo datos suficientes para ver los problemas que surjan.

Para obtener más información, consulte Configuración de reglas de muestreo de.

Encabezado de seguimiento

Todas las solicitudes se rastrean hasta alcanzar un mínimo configurable, después de lo cual se rastrea un porcentaje de solicitudes para evitar un costo innecesario. La decisión de muestreo y el ID de rastro se añaden a las solicitudes HTTP en los encabezados de rastreo denominados `X-Amzn-Trace-Id`. El primer servicio integrado en X-Ray, al que llega la solicitud añade un identificador de rastreo, que es leído por el SDK de X-Ray, e incluido en la respuesta.

Example Encabezado de rastreo con ID de rastro raíz y decisión de muestreo

```
X-Amzn-Trace-Id: Root=1-5759e988-
bd862e3fe1be46a994272793;Parent=53995c3f42cd8ad8;Sampled=1
```

> **Seguridad del encabezado de rastreo**
>
> Un encabezado de rastro se puede tener su origen en el SDK de X-Ray, un Servicio de AWS, o la solicitud del cliente. Su aplicación puede eliminar el encabezado `X-Amzn-Trace-Id` en

las solicitudes entrantes para evitar los problemas causados por usuarios que agregan ID de rastro o decisiones de muestreo a sus solicitudes.

El encabezado de rastreo también pueden contener un ID de segmento primario si la solicitud se originó a partir de una aplicación instrumentada. Por ejemplo, si su aplicación llama a una API web HTTP posterior con un cliente HTTP instrumentado, el SDK de X-Ray añade el ID del segmento de la solicitud original al encabezado de rastreo de la solicitud posterior. Una aplicación instrumentada que atiende la solicitud posterior puede registrar el ID del segmento primario para conectar ambas solicitudes.

Example Encabezado de rastreo con ID de rastro raíz, ID de segmento primario y decisión de muestreo

```
X-Amzn-Trace-Id: Root=1-5759e988-
bd862e3fe1be46a994272793;Parent=53995c3f42cd8ad8;Sampled=1
```

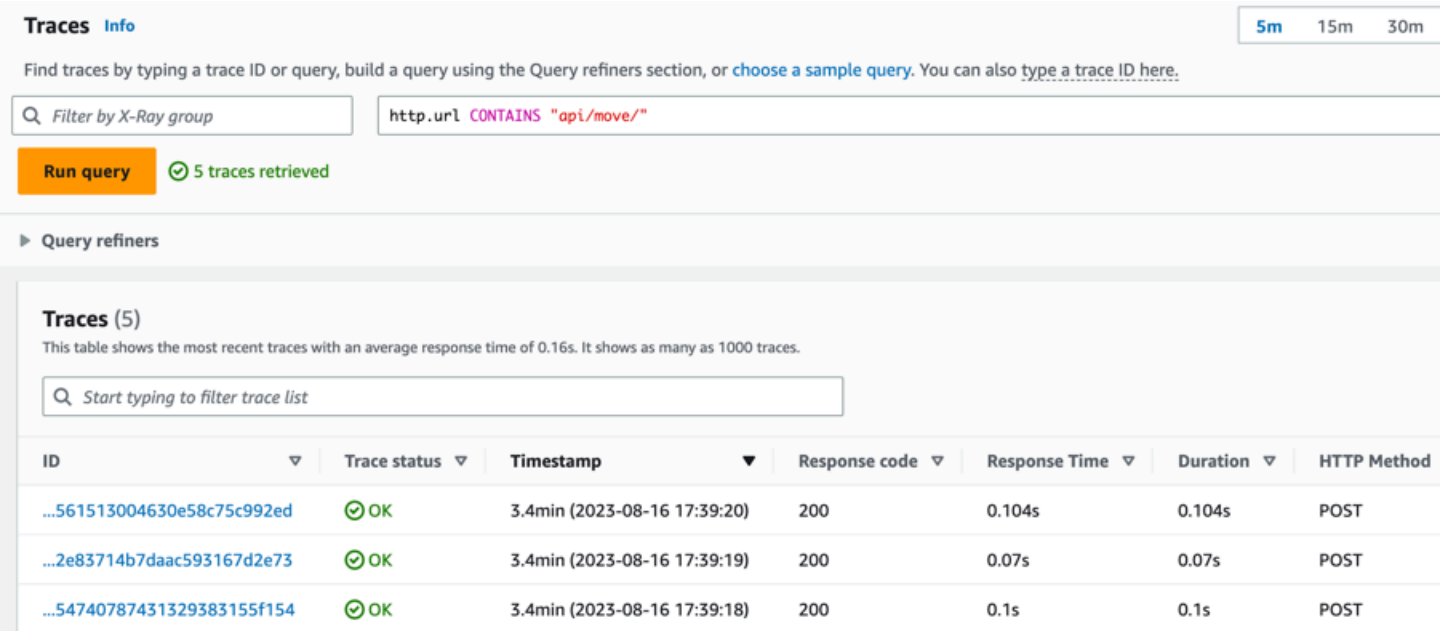
Lambda y otros Servicios de AWS pueden adjuntar `Lineage` al encabezado de rastro como parte de sus mecanismos de procesamiento y no debe usarse directamente.

Example Encabezado de rastreo con Lineage

```
X-Amzn-Trace-Id: Root=1-5759e988-
bd862e3fe1be46a994272793;Parent=53995c3f42cd8ad8;Sampled=1;Lineage=25:a87bd80c:1
```

Expresiones de filtro

Aunque se haya realizado un muestreo, una aplicación genera grandes cantidades de datos. La consola de AWS X-Ray ofrece una vista fácil de navegar del gráfico de servicio. Allí se muestra información sobre el estado y el rendimiento con la cual se puede identificar problemas y posibilidades de optimización en su aplicación. Cuando se realiza un rastreo avanzado, puede explorar en profundidad en busca de solicitudes individuales, o bien utilizar expresiones de filtro para encontrar rastros relacionados con rutas específicas o usuarios específicos.



Grupos

Como complemento de las expresiones de filtro, X-Ray también admite la característica de grupos. Mediante una expresión de filtro, puede definir los criterios para aceptar los rastros en el grupo.

Puede llamar al grupo por nombre o por nombre de recurso de Amazon (ARN) para generar su propio gráfico de servicios, resúmenes de rastros y métricas de Amazon CloudWatch. Una vez que se crea un grupo, los rastros de entrada se comparan con la expresión de filtro del grupo a medida que se almacenan en el servicio de X-Ray. Las métricas para el número de rastros que coinciden con cada criterio se publican en CloudWatch cada minuto.

La actualización de la expresión de filtro de un grupo no cambia los datos que ya se han registrado. La actualización se aplica únicamente a los rastros posteriores. Esto puede dar lugar a un gráfico que combina la expresión nueva con la anterior. Para evitarlo, elimine el grupo actual y cree uno nuevo.

> **Note**
>
> Los grupos se facturan por el número de rastros recuperados que coinciden con la expresión de filtro. Para más información, consulte Precios de AWS X-Ray.

Para obtener más información acerca de los grupos, consulte Configuración de grupos.

Anotaciones y metadatos

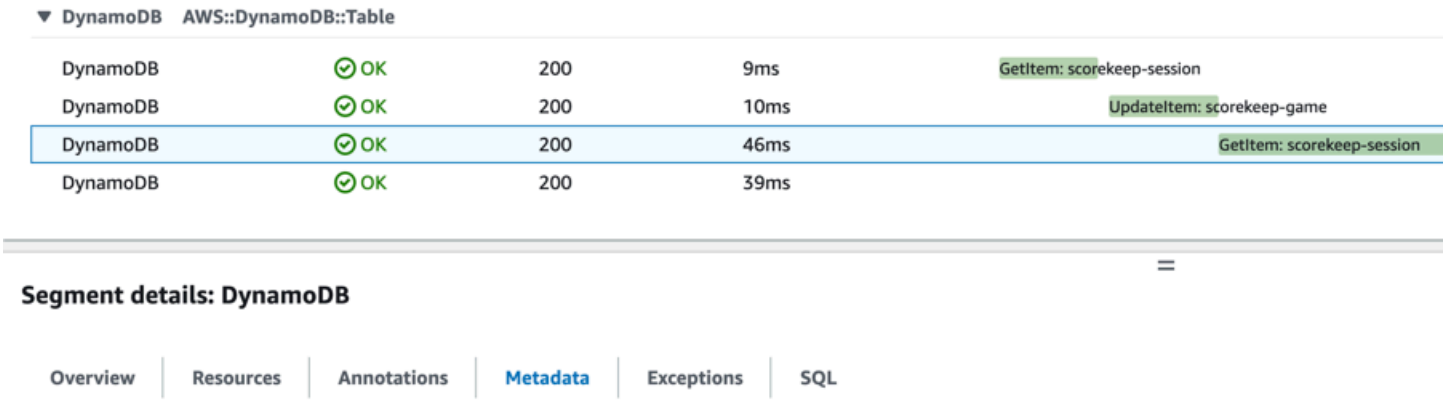
Cuando instrumenta su aplicación, el SDK de X-Ray registra información acerca de las solicitudes entrantes y de salida, los recursos de AWS utilizados y la aplicación en sí. Puede añadir más información al documento de segmento, por ejemplo anotaciones y metadatos. Las anotaciones y los metadatos se agregan en el nivel de seguimiento y se pueden añadir a cualquier segmento o subsegmento.

Las anotaciones son pares de clave-valor que se indexan para usarlos en expresiones de filtro. Utilice anotaciones para registrar los datos que desee utilizar para agrupar rastros en la consola o cuando llame a la API de `GetTraceSummaries`.

X-Ray indexa hasta 50 anotaciones por rastro.

Los metadatos son pares de clave-valor con valores de cualquier tipo, por ejemplo objetos y listas, pero que no se indexan. Utilice los metadatos para registrar los datos que desee almacenar en el rastro, pero que no vaya a usar para buscar rastros.

Puede ver anotaciones y metadatos en la ventana de los detalles de segmento o subsegmento, en la página Detalles de rastro de la consola de CloudWatch.



Errores y excepciones

X-Ray rastrea los errores que se producen en su código de aplicación y los errores que devuelven los servicios posteriores. Los errores se clasifican como se indica a continuación.

- **`Error`**: errores del cliente (errores de la serie 400)
- **`Fault`**: errores del servidor (errores de la serie 500)
- **`Throttle`**: errores de limitación controlada (429: demasiadas solicitudes)

Cuando se produce una excepción, mientras la aplicación está sirviendo una solicitud instrumentada, el SDK de X-Ray registra detalles sobre la excepción, incluido el rastro de pila, si está disponible. Puede ver excepciones en detalles de segmento en la consola de X-Ray.

Seguridad en AWS X-Ray

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que se ejecuta Servicios de AWS en la Nube de AWS. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener información sobre los programas de conformidad que se aplican a X-Ray, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por lo Servicio de AWS que utilice. Usted también es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación lo ayudará a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza X-Ray. En los siguientes temas, se le mostrará cómo configurar X-Ray para cumplir sus objetivos de seguridad y conformidad. También aprenderá a usar otros Servicios de AWS que pueden ayudarlo a monitorear y proteger sus recursos de X-Ray.

Temas

Protección de los datos en AWS X-Ray

AWS X-Ray siempre cifra los registros de seguimiento y los datos en reposo relacionados. Cuando necesite auditar y deshabilitar las claves de cifrado por motivos internos o de conformidad, puede

configurar X-Ray para que utilice una AWS Key Management Service (AWS KMS) a la hora de cifrar los datos.

X-Ray proporciona una Clave administrada de AWS denominada `aws/xray`. Utilice esta clave únicamente cuando desee auditar el uso de claves en AWS CloudTrail y no la propia clave. Cuando necesite administrar el acceso a la clave o configurar la rotación de claves, puede crear una clave administrada por el cliente.

Si cambia la configuración de cifrado, X-Ray tardará algún tiempo en generar y propagar las claves de datos. Aunque la nueva clave se esté procesando, X-Ray puede cifrar los datos utilizando la configuración anterior y la nueva de forma combinada. Si se modifica la configuración de cifrado, los datos existentes no volverán a cifrarse.

> **Note**
>
> AWS KMS genera costos cuando X-Ray utiliza una clave de KMS para cifrar o descifrar los datos de rastro.
>
> - Cifrado predeterminado: gratuito.
> - Clave administrada de AWS: se paga por el uso de las claves.
> - Clave administrada por el cliente: se paga por el uso y el almacenamiento de las claves.
>
> Para obtener más información, consulte AWS Key Management Service Pricing.

> **Note**
>
> Las notificaciones de información de X-Ray envían eventos a Amazon EventBridge, que actualmente no admite claves administradas por el cliente. Para obtener más información, consulte Protección de datos en Amazon EventBridge.

Debe tener acceso de nivel de usuario a una clave administrada por el cliente para configurar X-Ray, utilizarlo y, a continuación, consultar los rastros cifrados. Para obtener más información, consulte Permisos de usuario para cifrado.

CloudWatch console

Para configurar X-Ray de manera que utilice una clave de KMS para el cifrado mediante la consola de CloudWatch

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Configuración en el panel de navegación izquierdo.
3. Elija Ver ajustes en Cifrado dentro de la sección de Rastros de X-Ray.
4. Elija Editar en la sección Configuración del cifrado.
5. Elija Usar una clave de KMS.
6. Elija una clave en el menú desplegable:
   - aws/xray: utilice la Clave administrada de AWS.
   - Alias de clave: utilice una clave administrada por el cliente en su cuenta.
   - Especifique manualmente un ARN de clave: utilice una clave administrada por el cliente en una cuenta diferente. En el campo que aparece, escriba el nombre de recurso de Amazon (ARN) completo.
7. Elija Actualizar el cifrado.

X-Ray console

Para configurar X-Ray de manera que utilice una clave de KMS para el cifrado mediante la consola de X-Ray

1. Abra la consola de X-Ray.
2. Seleccione Encryption (Cifrado).
3. Elija Usar una clave de KMS.
4. Elija una clave en el menú desplegable:
   - aws/xray: utilice la Clave administrada de AWS.
   - Alias de clave: utilice una clave administrada por el cliente en su cuenta.
   - Especifique manualmente un ARN de clave: utilice una clave administrada por el cliente en una cuenta diferente. En el campo que aparece, escriba el nombre de recurso de Amazon (ARN) completo.

5. Seleccione Aplicar.

> **Note**
>
> X-Ray no es compatible con claves de KMS asimétricas.

Si X-Ray no puede obtener acceso a la clave de cifrado, deja de almacenar los datos. Esto puede ocurrir si el usuario pierde acceso a la clave de KMS o si se deshabilita una clave que actualmente está en uso. Cuando esto sucede, X-Ray muestra una notificación en la barra de navegación.

Para configurar las opciones de cifrado con la API de X-Ray, consulte Configuración de las opciones de muestreo, grupos y cifrado con la API de AWS X-Ray.

Administración de identidad y acceso para AWS X-Ray

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién puede estar autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de X-Ray. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades
- Administración del acceso con políticas
- Cómo funciona AWS X-Ray con IAM
- AWS X-Ray ejemplos de políticas basadas en la identidad
- Solución de problemas de identidades de AWS X-Ray y accesos

Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicita permisos a su administrador si no puede acceder a las características (consulte Solución de problemas de identidades de AWS X-Ray y accesos).
- Administrador del servicio: determina el acceso de los usuarios y envía solicitudes de permiso (consulte Cómo funciona AWS X-Ray con IAM).
- Administrador de IAM: escribe políticas para administrar el acceso (consulte AWS X-Ray ejemplos de políticas basadas en la identidad).

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en su Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear un Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz que tiene acceso completo a todos Servicios de AWS los recursos. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Para ver la lista completa de las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad que dispone de permisos específicos para una sola persona o aplicación. Recomendamos usar credenciales temporales en lugar de usuarios de IAM con credenciales a largo plazo. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica una colección de usuarios de IAM y facilita la administración de permisos para grandes conjuntos de usuarios. Para obtener más información, consulte Casos de usos para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Las funciones de IAM son útiles para el acceso de usuarios federados, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon. EC2 Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

El acceso se controla creando políticas y AWS adjuntándolas a identidades o recursos. AWS Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de política JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de política de permisos JSON que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden ser políticas insertadas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen políticas de confianza de roles de IAM y políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios compatibles. AWS WAF ACLs Para obtener más información ACLs, consulte la descripción general de la lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos otorgados por los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCPs): establece los permisos máximos disponibles para los recursos de tus cuentas. Para obtener más información, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.

- Políticas de sesión: políticas avanzadas que se transfieren como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona AWS X-Ray con IAM

Antes de utilizar IAM para administrar el acceso a X-Ray, debe comprender qué características de IAM están disponibles para su uso con X-Ray. Para obtener una perspectiva general sobre cómo funcionan X-Ray y otros Servicios de AWS con IAM, consulte Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Puede utilizar AWS Identity and Access Management (IAM) para conceder permisos para usar X-Ray a los usuarios y recursos informáticos de su cuenta. IAM controla el acceso al servicio X-Ray a nivel de la API para aplicar los permisos de manera uniforme, con independencia del cliente (consola, SDK de AWS o AWS CLI) que emplean los usuarios.

Para utilizar la consola de X-Ray para ver mapas de rastros y segmentos, solo necesita permisos de lectura. Para habilitar el acceso a la consola, añada la `AWSXrayReadOnlyAccess` política administrada al usuario de IAM.

Para desarrollo local y pruebas, cree un rol de IAM con permisos de lectura y escritura. Asuma el rol y almacene las credenciales temporales para el rol. Puede utilizar estas credenciales con el daemon de X-Ray, la AWS CLI y el SDK de AWS. Para obtener más información, consulte Uso de credenciales de seguridad temporales con AWS CLI.

Para implementar su aplicación instrumentada en AWS, cree un rol de IAM con permisos de escritura y asígneselo a los recursos que ejecutan la aplicación. `AWSXRayDaemonWriteAccess` incluye permisos para cargar rastros y algunos permisos de lectura, así como para admitir el uso de reglas de muestreo.

Las políticas de lectura y escritura no incluyen permiso para configurar la configuración de clave de cifrado y reglas de muestreo. Utilice `AWSXrayFullAccess` para obtener acceso a estas

configuraciones o añadir API de configuración en una política personalizada. Para el cifrado y el descifrado con una clave administrada por el cliente que cree, también necesita permiso para utilizar la clave.

Temas

- Políticas de X-Ray basadas en identidades
- Políticas de X-Ray basadas en recursos
- Autorización basada en etiquetas de X-Ray
- Ejecutar la aplicación de forma local
- Ejecutar la aplicación en AWS
- Permisos de usuario para cifrado

Políticas de X-Ray basadas en identidades

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. X-Ray admite acciones, claves de condición y recursos específicos. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte Referencia de los elementos de las políticas JSON de IAM en la Guía del usuario de IAM.

Acciones

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones de políticas de X-Ray utilizan el siguiente prefijo antes de la acción: `xray:`. Por ejemplo, para conceder a alguien permiso para recuperar detalles de los recursos de grupo con la operación de la API `GetGroup` de X-Ray, incluya la acción `xray:GetGroup` en su política. Las instrucciones de la política deben incluir un elemento `Action` o un elemento `NotAction`. X-Ray define su propio conjunto de acciones que describen las tareas que se pueden realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

```
"Action": [
      "xray:action1",
      "xray:action2"
```

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Get`, incluya la siguiente acción:

```
"Action": "xray:Get*"
```

Para ver una lista de las acciones de X-Ray, consulte Acciones definidas por AWS X-Ray en la Guía del usuario de IAM.

Recursos

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Para las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Puede controlar el acceso a los recursos a través de una política de IAM. Para las acciones que admiten permisos de nivel de recursos, se usa un nombre de recurso de Amazon (ARN) para identificar el recurso al que se aplica la política.

Es posible utilizar todas las acciones de X-Ray en una política de IAM para conceder o denegar permiso a los usuarios para utilizar esa acción. Sin embargo, no todas las acciones de X-Ray admiten permisos de nivel de recursos, que le permiten especificar los recursos en los que se puede realizar una acción.

Para las acciones que no admiten permisos de nivel de recursos, debe utilizar "*" como recurso.

Las siguientes acciones de X-Ray admiten permisos de nivel de recursos:

- `CreateGroup`
- `GetGroup`

- `UpdateGroup`
- `DeleteGroup`
- `CreateSamplingRule`
- `UpdateSamplingRule`
- `DeleteSamplingRule`

A continuación, se muestra un ejemplo de una política de permisos basada en identidad para una acción `CreateGroup`: El ejemplo muestra el uso de un ARN relacionado con el nombre de grupo `local-users` con el ID único como elemento comodín. El ID único se genera cuando se crea el grupo y, por lo tanto, no puede predecirse en la política con antelación. Cuando se utiliza `GetGroup`, `UpdateGroup` o `DeleteGroup`, puede definir esto como un elemento comodín o el ARN exacto, incluido el ID.

> **Note**
>
> El ARN de una regla de muestreo se define por su nombre. A diferencia de los ARN de grupo, las reglas de muestreo no tienen un ID generado de manera inequívoca.

Para ver una lista de tipos de recursos de X-Ray y sus ARN, consulte Recursos definidos por AWS X-Ray en la Guía del usuario de IAM. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por AWS X-Ray.

Claves de condición

X-Ray no proporciona ninguna clave de condición específica del servicio, pero sí admite el uso de algunas claves de condición globales. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWSen la Guía del usuario de IAM.

Ejemplos

Para ver ejemplos de políticas basadas en identidad de X-Ray, consulte AWS X-Ray ejemplos de políticas basadas en la identidad.

Políticas de X-Ray basadas en recursos

X-Ray admite políticas basadas en recursos para la integración actual y futura de Servicio de AWS, como el rastreo activo de Amazon SNS. Las políticas basadas en recursos de X-Ray se pueden

actualizar en otras Consola de administración de AWS o mediante el SDK o la CLI de AWS. Por ejemplo, la consola de Amazon SNS intenta configurar automáticamente una política basada en recursos para enviar rastros a X-Ray. En el siguiente documento de política se proporciona un ejemplo de configuración manual de una política basada en recursos de X-Ray.

Example Ejemplo de política basada en recursos de X-Ray para el rastreo activo de Amazon SNS

En este ejemplo de documento de política se especifican los permisos que Amazon SNS necesita para enviar datos de rastro a X-Ray:

```
{
    Version: "2012-10-17",
    Statement: [
      {
        Sid: "SNSAccess",
        Effect: Allow,
        Principal: {
          Service: "sns.amazonaws.com",
        },
        Action: [
          "xray:PutTraceSegments",
          "xray:GetSamplingRules",
          "xray:GetSamplingTargets"
        ],
        Resource: "*",
        Condition: {
          StringEquals: {
            "aws:SourceAccount": "account-id"
          },
          StringLike: {
            "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
          }
        }
      }
    ]
  }
```

Utilice la CLI para crear una política basada en recursos que dé a Amazon SNS permisos para enviar datos de rastro a X-Ray:

```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
 '{ "Version": "2012-10-17",        "Statement": [ { "Sid": "SNSAccess",
```

```
 "Effect": "Allow", "Principal": { "Service": "sns.amazonaws.com" }, "Action":
 [ "xray:PutTraceSegments", "xray:GetSamplingRules", "xray:GetSamplingTargets" ],
 "Resource": "*", "Condition": { "StringEquals": { "aws:SourceAccount": "account-
id" }, "StringLike": { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-
name" } } } ] }'
```

Para usar estos ejemplos, sustituya *partition*, *region*, *account-id* y *topic-name* por su partición de AWS, región, ID de cuenta y nombre de tema de Amazon SNS específicos. Para dar permiso a todos los temas de Amazon SNS para que envíen datos de rastro a X-Ray, sustituya el nombre del tema por *.

Autorización basada en etiquetas de X-Ray

Puede adjuntar etiquetas a los grupos o las reglas de muestreo de X-Ray, o pasar las etiquetas en una solicitud a X-Ray. Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `xray:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Para obtener más información acerca del etiquetado de recursos de X-Ray, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Gestión del acceso a los grupos de rayos X y a las reglas de muestreo en función de las etiquetas.

Ejecutar la aplicación de forma local

Su aplicación instrumentada envía datos de rastro al daemon de X-Ray. El daemon almacena en búfer documentos de segmento y los carga en lotes en el servicio X-Ray. El daemon necesita permisos de escritura para cargar los datos de rastro y telemetría en el servicio X-Ray.

Al ejecutar el daemon de forma local, se crea un rol de IAM, se asume el rol y se almacenan las credenciales temporales en variables de entorno o en un archivo denominado `credentials` dentro de una carpeta denominada `.aws` en la carpeta de usuario. Para obtener más información, consulte Uso de credenciales de seguridad temporales con AWS CLI.

Example~/.aws/credentials

```
[default]
aws_access_key_id={access key ID}
aws_secret_access_key={access key}
```

```
aws_session_token={AWS session token}
```

Si ya ha configurado credenciales para usarlas con el SDK o la AWS CLI de AWS, el daemon puede utilizarlas. En caso de que haya varios perfiles disponibles, el daemon utilizará el perfil predeterminado.

Ejecutar la aplicación en AWS

Cuando ejecute su aplicación en AWS, utilice un rol para conceder permisos a la instancia de Amazon EC2 o a la función de Lambda que ejecute el daemon.

- Amazon Elastic Compute Cloud (Amazon EC2): cree un rol de IAM y adjúntelo a la instancia de EC2 como perfil de instancia.
- Amazon Elastic Container Service (Amazon ECS): cree un rol de IAM y adjúntelo a las instancias de contenedor como rol de IAM de instancia de contenedor.
- AWS Elastic Beanstalk(Elastic Beanstalk): Elastic Beanstalk incluye los permisos de X-Ray en su perfil de instancia predeterminado. Puede usar el perfil de instancia predeterminado o añadir permisos de escritura a un perfil de instancia personalizado.
- AWS Lambda (Lambda): añada permisos de escritura al rol de ejecución de la función.

Para crear un rol y usarlo con X-Ray

1. Abra la consola de IAM.
2. Elija Roles.
3. Elija Crear nuevo rol.
4. En Role Name (Nombre del rol), escriba **`xray-application`**. Elija Paso siguiente.
5. En Role Type (Tipo de rol), elija Amazon EC2.
6. Adjunte la siguiente política administrada para que la aplicación tenga acceso a los Servicios de AWS:

   - AWSXRayDaemonWriteAccess: da permiso al daemon de X-Ray para cargar datos de rastro.

   Si la aplicación utiliza el SDK de AWS para acceder a otros servicios, añada políticas que otorguen acceso a dichos servicios.
7. Elija Paso siguiente.
8. Seleccione Crear rol.

Permisos de usuario para cifrado

X-Ray cifra todos los datos de rastro y de forma predeterminada y puede configurarlo para que use una clave que usted administre. Si elige una clave administrada por el cliente de AWS Key Management Service, tiene que asegurarse de que la política de acceso a la clave le conceda permisos a X-Ray para utilizarla para cifrar. Otros usuarios de su cuenta también tienen que obtener acceso a la clave para ver los datos de rastro cifrados en la consola de X-Ray.

Para una clave administrada por el cliente, configure su clave con una política de acceso que permita las siguientes acciones:

- El usuario que configura la clave en X-Ray tiene permisos para llamar a `kms:CreateGrant` y `kms:DescribeKey`.
- Los usuarios que pueden tener acceso a los datos de rastreo cifrados tienen permiso para llamar a `kms:Decrypt`.

Cuando se añade un usuario al grupo Usuarios clave en la sección de configuración de clave de la consola de , tienen permisos para ambas operaciones. El permiso solo se tiene que establecer en la política de claves, por tanto no necesita ningún permiso de AWS KMS en sus usuarios, grupos o roles. Consulte Uso de políticas de claves en la Guía para desarrolladores de AWS KMS.

Para el cifrado predeterminado o si elige la CMK administrada de AWS (`aws/xray`), el permiso se basa en quién tiene acceso a las API de X-Ray. Cualquier persona con acceso a `PutEncryptionConfig`, incluido en `AWSXrayFullAccess`, puede cambiar la configuración de cifrado. Para evitar que un usuario cambie la clave de cifrado, no le conceda permiso para utilizar `PutEncryptionConfig`.

AWS X-Ray ejemplos de políticas basadas en la identidad

De forma predeterminada, los usuarios y roles no tienen permiso para crear, ver ni modificar recursos de X-Ray. Tampoco pueden realizar tareas con la API Consola de administración de AWS AWS CLI, o AWS . Un administrador debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe adjuntar esas políticas a los usuarios o grupos que necesiten esos permisos.

Para obtener información acerca de cómo crear una política basada en identidad de IAM con estos documentos de políticas JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Temas

Prácticas recomendadas relativas a políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de X-Ray de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos en muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Uso de la consola de X-Ray

Para acceder a la AWS X-Ray consola, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los recursos de X-Ray de su propiedad Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

Para garantizar que esas entidades puedan seguir utilizando la consola de X-Ray, adjunte la política `AWSXRayReadOnlyAccess` AWS gestionada a las entidades. Esta política se describe con más detalle en las políticas administradas por IAM para X-Ray. Para obtener más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM.

No es necesario conceder permisos mínimos de consola a los usuarios que solo realizan llamadas a la API AWS CLI o a la AWS API. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intenta realizar.

Permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Gestión del acceso a los grupos de rayos X y a las reglas de muestreo en función de las etiquetas

Puede utilizar las condiciones de su política basada en la identidad para controlar el acceso a los grupos y reglas de muestreo de X-Ray basados en etiquetas. El siguiente ejemplo de política podría utilizarse para denegar a un rol de usuario los permisos para crear, eliminar o actualizar grupos con las etiquetas `stage:prod` `ostage:preprod`. Para obtener más información sobre el etiquetado de

las reglas y grupos de muestreo de rayos X, consulte Etiquetado de reglas de muestreo y grupos de X-Ray.

Para denegar la creación de una regla de muestreo, utilice esta `aws:RequestTag` opción para indicar las etiquetas que no se pueden transferir como parte de una solicitud de creación. Para denegar la actualización o la eliminación de una regla de muestreo, utilice esta `aws:ResourceTag` opción para denegar las acciones basadas en las etiquetas de esos recursos.

Puede adjuntar estas políticas (o combinarlas en una sola política y, a continuación, adjuntar la política) a los usuarios de su cuenta. Para que el usuario realice cambios en un grupo o regla de muestreo, el grupo o la regla de muestreo no debe estar etiquetado `stage=prepod` o `stage=prod`. La clave de la etiqueta de condición `Stage` coincide con los nombres de las claves de condición `Stage` y `stage` porque no distinguen entre mayúsculas y minúsculas. Para obtener más información acerca de las claves de condición, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.

Un usuario con un rol que tenga la siguiente política adjunta no puede agregar la etiqueta `role:admin` a los recursos ni eliminar etiquetas de un recurso que esté `role:admin` asociada a ella.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllXRay",
            "Effect": "Allow",
            "Action": "xray:*",
            "Resource": "*"
        },
        {
            "Sid": "DenyRequestTagAdmin",
            "Effect": "Deny",
            "Action": "xray:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/role": "admin"
                }
            }
```

```
        },
        {
            "Sid": "DenyResourceTagAdmin",
            "Effect": "Deny",
            "Action": "xray:UntagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/role": "admin"
                }
            }
        }
    ]
}
```

Políticas administradas por IAM para X-Ray

A fin de facilitar la concesión de permisos, IAM admite políticas administradas en cada servicio. Un servicio puede actualizar estas políticas administradas con nuevos permisos cuando publique nuevos permisos. APIs AWS X-Ray proporciona políticas administradas para casos de uso de solo lectura, solo de escritura y de administrador.

- `AWSXrayReadOnlyAccess`— Lea los permisos para usar la consola de X-Ray o el AWS SDK para obtener datos de rastreo, mapas de rastreo, información y configuración de X-Ray desde la API de X-Ray. AWS CLI Incluye Observability Access Manager (OAM) `oam:ListSinks` y `oam:ListAttachedSinks` permisos que permiten a la consola ver los rastreos compartidos desde las cuentas de origen como parte de la observabilidad CloudWatch entre cuentas. Las acciones `BatchGetTraceSummaryById` y de la `GetDistinctTraceGraphs` API no están diseñadas para que el código las invoque y, por lo tanto, no están incluidas en el comando y. AWS CLI AWS SDKs

  ```
  {
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                  "xray:GetSamplingRules",
                  "xray:GetSamplingTargets",
                  "xray:GetSamplingStatisticSummaries",
  ```

```
                "xray:BatchGetTraces",
                "xray:BatchGetTraceSummaryById",
                "xray:GetDistinctTraceGraphs",
                "xray:GetServiceGraph",
                "xray:GetTraceGraph",
                "xray:GetTraceSummaries",
                "xray:GetGroups",
                "xray:GetGroup",
                "xray:ListTagsForResource",
                "xray:ListResourcePolicies",
                "xray:GetTimeSeriesServiceStatistics",
                "xray:GetInsightSummaries",
                "xray:GetInsight",
                "xray:GetInsightEvents",
                "xray:GetInsightImpactGraph",
                "oam:ListSinks"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }

}
```

- `AWSXRayDaemonWriteAccess`— Permisos de escritura para usar el daemon de X-Ray o el AWS SDK para cargar documentos de segmentos y telemetría a la API de X-Ray. AWS CLI Incluye permisos de lectura para obtener reglas de muestreo y notificar los resultados de muestreo.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
```

```
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- `AWSXrayCrossAccountSharingConfiguration`: otorga permisos para crear, administrar y ver los enlaces de Observability Access Manager para compartir los recursos de X-Ray entre cuentas. Se usa para permitir la observabilidad CloudWatch entre cuentas entre cuentas de origen y de monitoreo.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:Link",
                "oam:ListLinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
```

```
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        }
    ]

}
```

- `AWSXrayFullAccess`— Permiso para usar todos los X-Ray APIs, incluidos los permisos de lectura, los permisos de escritura y el permiso para configurar los ajustes de las claves de cifrado y las reglas de muestreo. Incluye el administrador de acceso a la observabilidad (OAM) `oam:ListSinks` y `oam:ListAttachedSinks` permisos que permiten a la consola ver las trazas compartidas desde las cuentas de origen como parte de la observabilidad CloudWatch entre cuentas.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:*",
                "oam:ListSinks"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

Para añadir una política administrada a un usuario, un grupo o un rol de IAM

1. Abra la consola de IAM.
2. Abra el rol asociado a su perfil de instancia, al usuario de IAM o al grupo de usuarios de IAM.
3. En Permissions (Permisos), asocia la política administrada.

Actualizaciones de X-Ray a las políticas AWS gestionadas

Consulta los detalles sobre las actualizaciones de las políticas AWS gestionadas de X-Ray desde que este servicio comenzó a rastrear estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página del historial de documentos de X-Ray.

Cambio	Descripción	Fecha
Políticas administradas por IAM para X-Ray: se agregaron nuevas políticas `AWSXrayCrossAccountSharingConfiguration` y se actualizaron las políticas `AWSXrayReadOnlyAccess` y `AWSXrayFullAccess`.	X-Ray agregó permisos de Observability Access Manager (OAM) `oam:ListSinks` y `oam:ListAttachedSinks` a estas políticas para permitir que la consola vea los rastros compartidos desde las cuentas de origen como parte de la observabilidad CloudWatch entre cuentas.	27 de noviembre de 2022
Políticas administradas por IAM para X-Ray: actualización de la política `AWSXrayReadOnlyAccess`.	X-Ray agregó una acción de API, `ListResourcePolicies`.	15 de noviembre de 2022
Uso de la consola X-Ray: actualización de la política `AWSXrayReadOnlyAccess`	X-Ray agregó dos nuevas acciones de API `BatchGetTraceSummaryById` y `GetDistinctTraceGraphs`.	11 de noviembre de 2022

Cambio	Descripción	Fecha
	Estas acciones API no se han diseñado para que se llamen desde el código. Por lo tanto, estas acciones de la API no están incluidas en la sección y. AWS CLI AWS SDKs	

Especificación de un recurso en una política de IAM

Puede controlar el acceso a los recursos a través de una política de IAM. Para las acciones que admiten permisos de nivel de recursos, se usa un nombre de recurso de Amazon (ARN) para identificar el recurso al que se aplica la política.

Es posible utilizar todas las acciones de X-Ray en una política de IAM para conceder o denegar permiso a los usuarios para utilizar esa acción. Sin embargo, no todas las acciones de X-Ray admiten permisos de nivel de recursos, que le permiten especificar los recursos en los que se puede realizar una acción.

Para las acciones que no admiten permisos de nivel de recursos, debe utilizar "`*`" como recurso.

Las siguientes acciones de X-Ray admiten permisos de nivel de recursos:

- `CreateGroup`
- `GetGroup`
- `UpdateGroup`
- `DeleteGroup`
- `CreateSamplingRule`
- `UpdateSamplingRule`
- `DeleteSamplingRule`

A continuación, se muestra un ejemplo de una política de permisos basada en identidad para una acción `CreateGroup`: El ejemplo muestra el uso de un ARN relacionado con el nombre de grupo `local-users` con el ID único como elemento comodín. El ID único se genera cuando se crea el grupo y, por lo tanto, no puede predecirse en la política con antelación. Cuando se utiliza `GetGroup`,

`UpdateGroup` o `DeleteGroup`, puede definir esto como un elemento comodín o el ARN exacto, incluido el ID.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateGroup"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
            ]
        }
    ]
}
```

A continuación, se muestra un ejemplo de una política de permisos basada en identidad para una acción `CreateSamplingRule`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateSamplingRule"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:sampling-rule/base-
scorekeep"
            ]
        }
    ]
}
```

> **Note**
>
> El ARN de una regla de muestreo se define por su nombre. A diferencia del grupo ARNs, las reglas de muestreo no tienen un ID generado de forma única.

Solución de problemas de identidades de AWS X-Ray y accesos

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con X-Ray e IAM.

Temas

No tengo autorización para realizar una acción en X-Ray.

Si la Consola de administración de AWS le indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su gestionador para recibir ayuda. El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

En el siguiente ejemplo, el error se produce cuando el usuario de `mateojackson` intenta utilizar la consola para ver detalles sobre una regla de muestreo, pero no tiene permisos `xray:GetSamplingRules`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: xray:GetSamplingRules on resource: arn:${Partition}:xray:${Region}:
${Account}:sampling-rule/${SamplingRuleName}
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso de regla de muestreo mediante la acción `xray:GetSamplingRules`.

No tengo autorización para realizar la acción iam:PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a X-Ray.

Algunos Servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en X-Ray. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su gestionador de AWS. El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

Soy administrador y deseo permitir que otros obtengan acceso a X-Ray.

Para permitir que otras personas accedan a X-Ray, debe conceder permiso a las personas o aplicaciones que lo necesiten. Si usa AWS IAM Identity Center para administrar las personas y las aplicaciones, debe asignar conjuntos de permisos a los usuarios o grupos para definir su nivel de acceso. Los conjuntos de permisos crean políticas de IAM y las asignan a los roles de IAM asociados a la persona o aplicación de forma automática. Para obtener más información, consulte la sección Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center.

Si no utiliza IAM Identity Center, debe crear entidades de IAM (usuarios o roles) para las personas o aplicaciones que necesitan acceso. A continuación, debe adjuntar una política a la entidad que le conceda los permisos correctos en X-Ray. Una vez concedidos los permisos, proporcione las credenciales al usuario o al desarrollador de la aplicación. Utilizarán esas credenciales para acceder a AWS. Para obtener más información sobre la creación de usuarios, grupos, políticas y permisos de IAM, consulte Identidades de IAM y Políticas y permisos en IAM en la Guía del usuario de IAM.

Quiero permitir a personas externas a mi Cuenta de AWS el acceso a mis recursos de X-Ray.

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que

asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para obtener información acerca de si X-Ray admite estas características, consulte Cómo funciona AWS X-Ray con IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuentas de AWS de su propiedad, consulte Acceso para un usuario de IAM en otra Cuenta de AWS propia en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a tus recursos a Cuentas de AWS de terceros, consulta Proporcionar acceso a Cuentas de AWS que son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulta Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Registro y monitorización en AWS X-Ray

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de sus soluciones de AWS. Debe recopilar datos de supervisión de todas las partes de su solución de AWS para que pueda depurar un error multipunto de una forma más fácil si se produce. AWS proporciona varias herramientas para supervisar sus recursos de X-Ray y responder a posibles incidentes:

Registros de AWS CloudTrail

> AWS X-Ray se integra con AWS CloudTrail para registrar las acciones de la API realizadas por un usuario, un rol o un servicio de AWS en X-Ray. Puede utilizar CloudTrail para supervisar las solicitudes de la API de X-Ray en tiempo real y almacenar registros en Amazon S3, Registros de Amazon CloudWatch y Eventos de Amazon CloudWatch. Para obtener más información, consulte Registro de llamadas a la API de X-Ray con AWS CloudTrail.

AWS ConfigSeguimiento de

AWS X-Ray se integra en AWS Config para registrar cambios de configuración realizados en los recursos de cifrado de X-Ray. Puede utilizar AWS Config para realizar un inventario de recursos de cifrado de X-Ray, auditar el historial de configuración de X-Ray y enviar notificaciones basadas en los cambios de recursos. Para obtener más información, consulte Rastreo de los cambios en la configuración de cifrado de X-Ray con AWS Config.

Supervisión de Amazon CloudWatch

Puede usar el SDK de X-Ray para Java con el fin de publicar métricas de Amazon CloudWatch sin muestrear de los segmentos de X-Ray recopilados. Estas métricas se derivan de la hora de inicio y finalización del segmento, y los marcadores de estado limitado, fallo y error. Utilice estas métricas de seguimiento para exponer los reintentos y los problemas de dependencia con los subsegmentos. Para obtener más información, consulte AWS X-Ray métricas del X-Ray SDK para Java.

Validación de conformidad en AWS X-Ray

Para saber si un Servicio de AWS está incluido en el ámbito de programas de conformidad específicos, consulte Servicios de AWS incluidos por programa de conformidad y escoja el programa de conformidad que le interese. Para obtener información general, consulte Programas de conformidad de AWS.

Puedes descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Servicios de AWS se determina en función de la confidencialidad de los datos, los objetivos de conformidad de su empresa, así como de la legislación y los reglamentos aplicables. Para obtener más información sobre la responsabilidad de conformidad al usar Servicios de AWS, consulte la Documentación de seguridad de AWS.

Resiliencia en AWS X-Ray

La infraestructura global de AWS se divide en Regiones de AWS y zonas de disponibilidad. Las Regiones de AWS proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas

de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las Regiones de AWS y las zonas de disponibilidad, consulte la Infraestructura global de AWS.

Seguridad de la infraestructura en () AWS X-Ray

Como se trata de un servicio administrado, AWS X-Ray está protegido por la seguridad de red global de AWS. Para obtener información sobre los servicios de seguridad de AWS y cómo AWSprotege la infraestructura, consulte Seguridad en la nube de AWS. Para diseñar su entorno de AWS con las prácticas recomendadas de seguridad de la infraestructura, consulte Protección de la infraestructura en Portal de seguridad de AWS Well-Architected Framework.

Puede utilizar llamadas a API publicadas por AWS para acceder a X-Ray a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Uso de AWS X-Ray con puntos de conexión de la VPC

Si utiliza Amazon Virtual Private Cloud (Amazon VPC) para alojar sus recursos de AWS, puede establecer una conexión privada entre su VPC y X-Ray. Eso habilitará recursos en su Amazon VPC para comunicarse con el servicio de X-Ray sin pasar por la Internet pública.

Amazon VPC es un Servicio de AWS que puede utilizar para lanzar recursos de AWS en una red virtual que usted defina. Con una VPC, puede controlar la configuración de la red, como el rango de direcciones IP, las subredes, las tablas de enrutamiento y las puertas de enlace de red. Para conectar una VPC a X-Ray, debe definir un punto de conexión de VPC de tipo interfaz. El punto de conexión ofrece conectividad escalable de confianza con X-Ray sin necesidad de utilizar una puerta de enlace de Internet, una instancia de traducción de direcciones de red (NAT) o una conexión de VPN. Para obtener más información, consulte ¿Qué es Amazon VPC?) en la Guía del usuario de Amazon VPC.

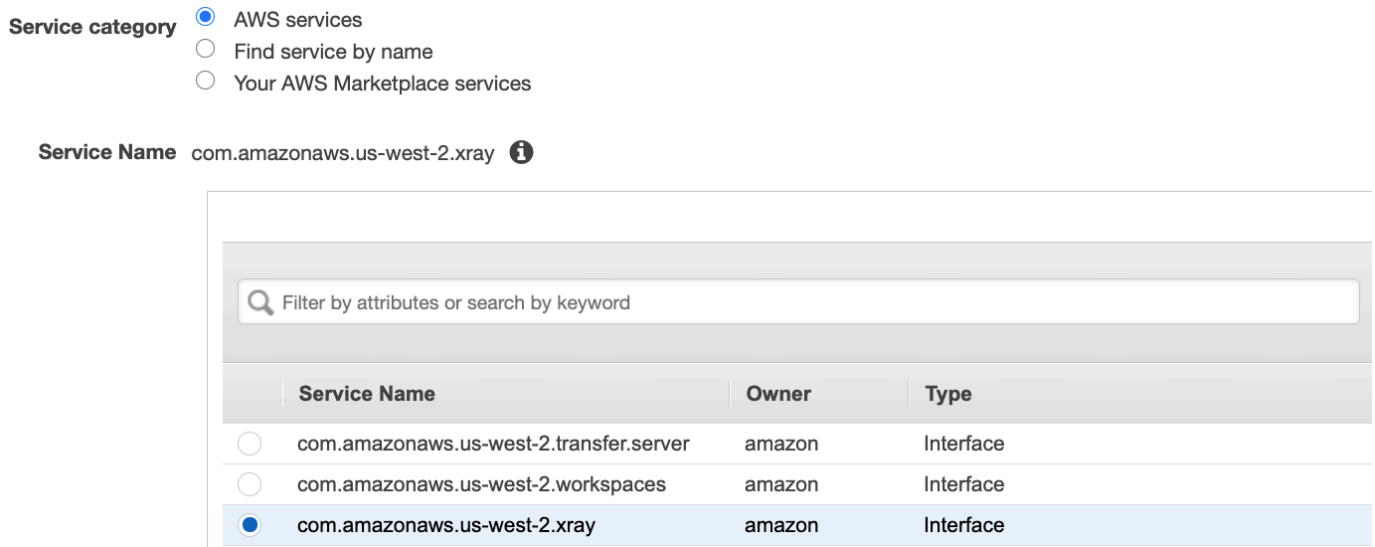
Los puntos de conexión de la VPC de tipo interfaz utilizan la tecnología de AWS PrivateLink, una tecnología de AWS que permite la comunicación privada entre los Servicios de AWS mediante una interfaz de red elástica con direcciones IP privadas. Para obtener más información, consulte la publicación del blog New – AWS PrivateLink for Servicios de AWS y Getting Started en la Guía del usuario de Amazon VPC.

Para asegurarse de que puede crear un punto de conexión de VPC para X-Ray en la Región de AWS que usted desea, consulte Regiones admitidas.

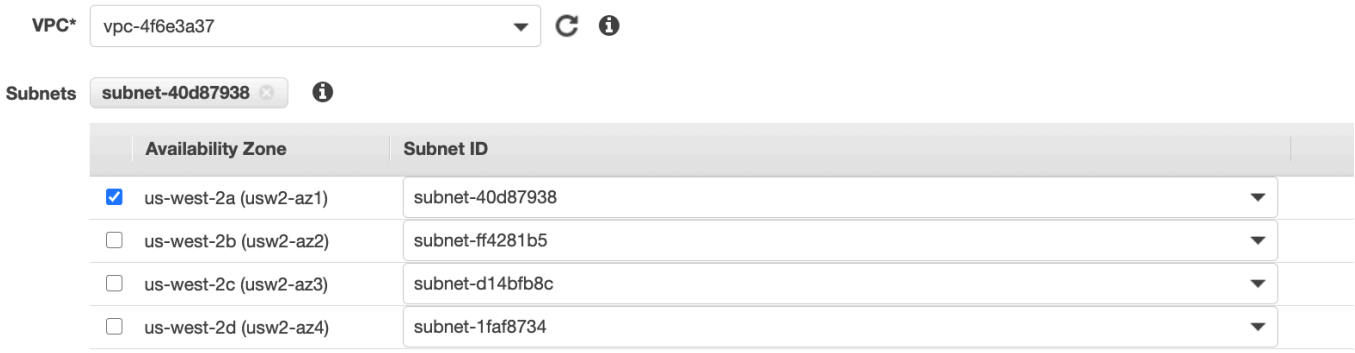
Creación de un punto de conexión de VPC para X-Ray

Para comenzar a utilizar X-Ray con su VPC, cree un punto de conexión de VPC de tipo interfaz para X-Ray.

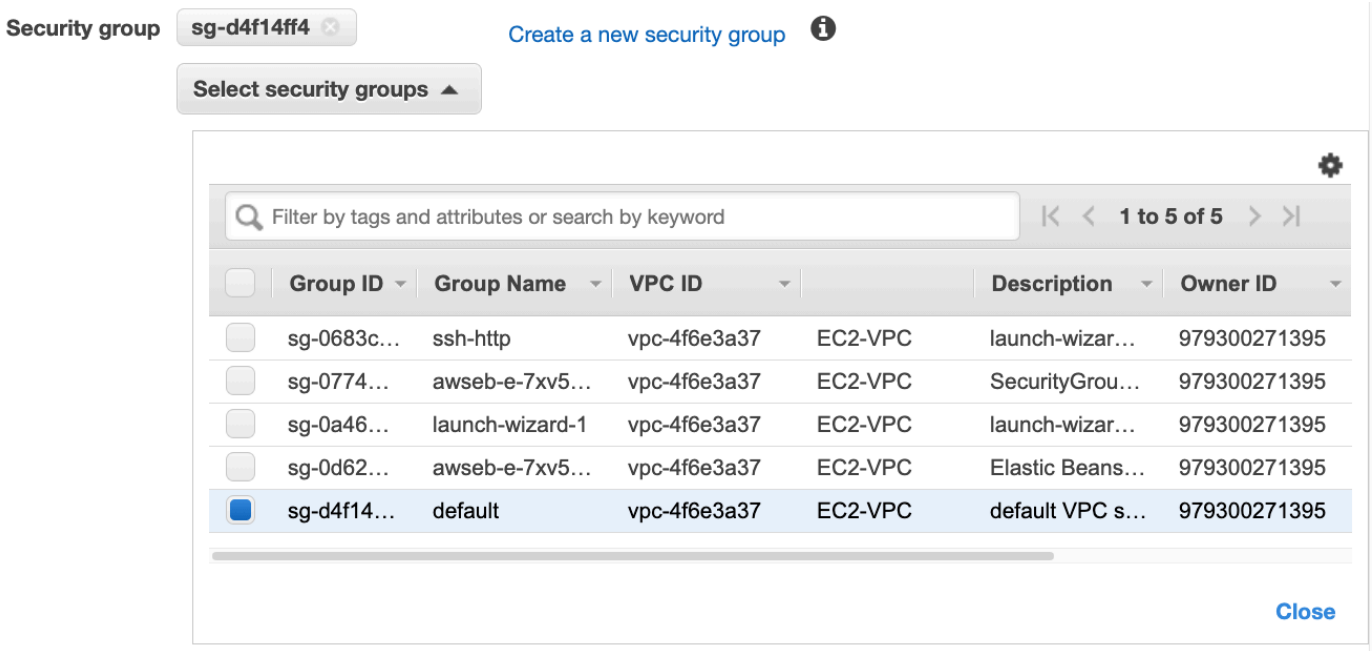
1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. Navegue hasta los puntos de conexión en el panel de navegación y seleccione Crear punto de conexión.
3. Busque y seleccione el nombre del servicio de AWS X-Ray: `com.amazonaws.region.xray`.



4. Seleccione la VPC que desee y, a continuación, seleccione una subred de la VPC para utilizar el punto de conexión de tipo interfaz. Se creará una interfaz de red de punto de conexión en la subred seleccionada. Puede especificar más de una subred en zonas de disponibilidad distintas (si el servicio lo admite) para asegurarse de que el punto de conexión de interfaz sea resistente a errores de zonas de disponibilidad. Si lo hace, se crea una interfaz de red en cada subred que especifique.



5. (Opcional) El DNS privado está habilitado de forma predeterminada para el punto de conexión, de modo que pueda realizar solicitudes a X-Ray mediante el nombre de host de DNS predeterminado. Si lo desea, puede deshabilitarlo.
6. Especificar los grupos de seguridad que se asociarán a la interfaz de red de punto de conexión.



7. (Opcional) Especifique una política personalizada para controlar los permisos de acceso al servicio de X-Ray. De forma predeterminada, se permite totalmente el acceso.

Control del acceso al punto de conexión de VPC de X-Ray

Una política de punto de conexión de VPC es una política de recursos de IAM que puede asociar a un punto de conexión cuando crea o modifica el punto de conexión. Si no adjunta una política al crear un punto de enlace, Amazon VPC adjunta una política predeterminada que le conceda acceso completo al servicio. Una política de punto de enlace no anula ni sustituye a las políticas de usuario

de IAM ni las políticas específicas del servicio. Se trata de una política independiente para controlar el acceso desde el punto de conexión al servicio especificado. Las políticas de punto de conexión deben escribirse en formato JSON. Para obtener más información, consulte Controlar el acceso a servicios con puntos de conexión de VPC en la Guía del usuario de Amazon VPC.

La política de puntos de conexión de VPC le permite controlar los permisos de varias acciones de X-Ray. Por ejemplo, puede crear una política para permitir solo PutTraceSegment y denegar el resto de las acciones. Eso restringe las cargas de trabajo y los servicios en la VPC para que solo se envíen datos de rastro a X-Ray y se deniegue cualquier otra acción, como recuperar datos, cambiar la configuración de cifrado o crear o actualizar grupos.

A continuación, se muestra un ejemplo de una política de puntos de conexión de X-Ray. Esta política permite a los usuarios que se conectan a X-Ray través de la VPC enviar datos de segmentos a X-Ray y les impide realizar otras acciones de X-Ray.

```
{"Statement": [
    {"Sid": "Allow PutTraceSegments",
      "Principal": "*",
      "Action": [
        "xray:PutTraceSegments"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Para editar la política de punto de conexión de VPC para X-Ray

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación, elija Puntos de conexión.
3. Si aún no ha creado el punto de conexión para X-Ray, siga los pasos descritos en Creación de un punto de conexión de VPC para X-Ray.
4. Seleccione el punto de conexión com.amazonaws.***region***.monitoring y, a continuación, elija la pestaña Política.
5. Elija Editar política y, a continuación, realice los cambios.

Regiones admitidas

Actualmente X-Ray admite puntos de conexión de la VPC en las siguientes regiones de Regiones de AWS:

- Este de EE. UU. (Ohio)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- África (Ciudad del Cabo)
- Asia-Pacífico (Hong Kong)
- Asia-Pacífico (Bombay)
- Asia-Pacífico (Osaka)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- Asia-Pacífico (Tokio)
- Canadá (centro)
- Europa (Fráncfort)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milán)
- Europa (París)
- Europa (Estocolmo)
- Medio Oriente (Baréin)
- América del Sur (São Paulo)
- AWS GovCloud (Este de EE. UU.)
- AWS GovCloud (Oeste de EE. UU.)

Prevención de la sustitución confusa entre servicios

El problema de la sustitución confusa es un problema de seguridad en el que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación entre servicios puede dar lugar al problema de la sustitución confusa. La suplantación entre servicios puedes producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puedes manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que lo ayudan a proteger sus datos para todos los servicios con entidades principales de servicio a las que se les ha dado acceso a los recursos de su cuenta.

Se recomienda utilizar las claves de contexto de condición global `aws:SourceArn`, `aws:SourceAccount`, `aws:SourceOrgID` y `aws:SourceOrgPaths` en las políticas de recursos para limitar los permisos que concede xraylong a otro servicio para el recurso. Utilice `aws:SourceArn` para asociar solo un recurso al acceso entre servicios. Utilice `aws:SourceAccount` para permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios. Utilice `aws:SourceOrgID` para permitir que cualquier recurso de cuentas dentro de una organización se asocie al uso entre servicios. Utilice `aws:SourceOrgPaths` para asociar cualquier recurso de cuentas dentro de una ruta de AWS Organizations al uso entre servicios. Para obtener más información acerca de las rutas de acceso, consulte Comprender la ruta de la entidad de AWS Organizations.

La forma más eficaz de protegerse contra el problema de la sustitución confusa es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso. Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de condición de contexto global `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`.

Si el valor de `aws:SourceArn` no contiene el ID de cuenta, como un ARN de bucket de Amazon S3, debe utilizar `aws:SourceAccount` y `aws:SourceArn` para limitar los permisos.

Para protegerse contra el problema del suplente confuso a gran escala, utilice la clave de contexto de condición global `aws:SourceOrgID` o `aws:SourceOrgPaths`con el identificador de organización o la ruta de organización del recurso en sus políticas basadas en recursos. Las políticas que incluyan la clave `aws:SourceOrgID` o `aws:SourceOrgPaths` incluirán automáticamente las cuentas correctas y no requerirán una actualización manual cuando se agregan, quitan o mueven cuentas en la organización.

El siguiente ejemplo muestra cómo se pueden utilizar las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en xray para prevenir el error de la sustitución confusa.

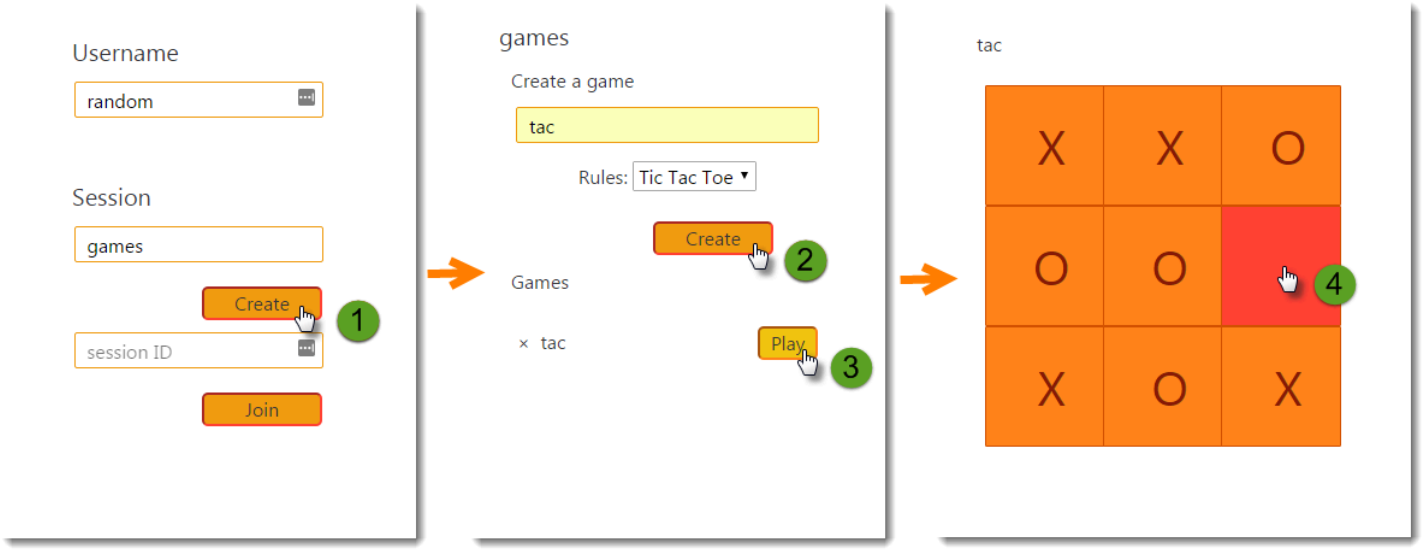
```
{
      "Sid": "BlockCrossAccountUnlessSameSource",
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKeyWithoutPlaintext"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": "123456789012",
          "aws:SourceAccount": "123456789012"
        },
        "ArnNotLike": {
          "aws:SourceArn": "arn:*:*:*:123456789012:*"
        }
      }
    }
```

AWS X-Ray ejemplo de aplicación

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

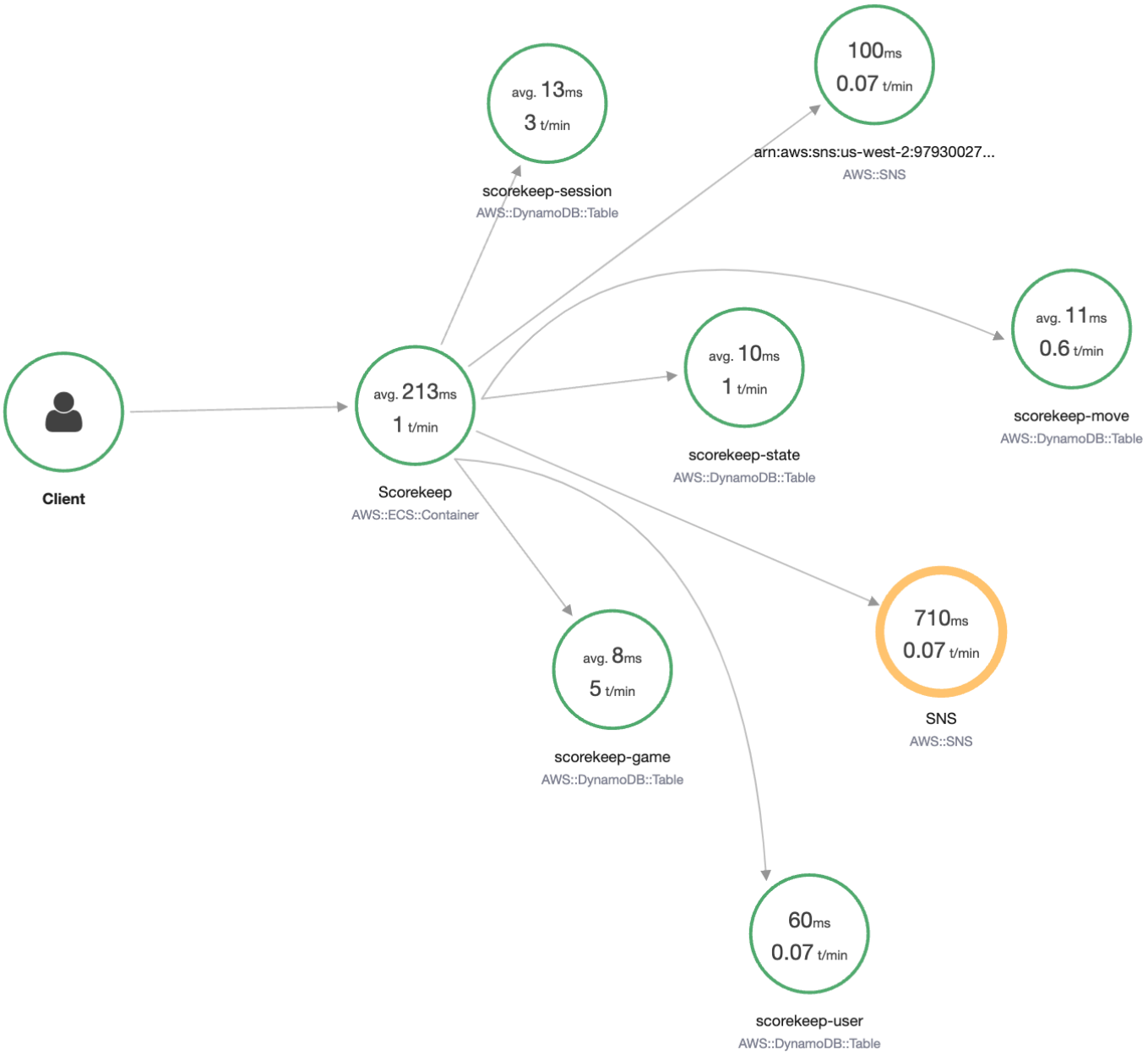
La aplicación de eb-java-scorekeepejemplo AWS X-Ray, disponible en GitHub, muestra el uso del AWS X-Ray SDK para instrumentar las llamadas HTTP entrantes, los clientes del SDK de DynamoDB y los clientes HTTP. La aplicación de ejemplo se utiliza CloudFormation para crear tablas de DynamoDB, compilar código Java en una instancia y ejecutar el daemon X-Ray sin ninguna configuración adicional.

Consulte el tutorial de Scorekeep para empezar a instalar y utilizar una aplicación de ejemplo instrumentada, utilizando el o el. Consola de administración de AWS AWS CLI



La muestra incluye una aplicación web frontend, la API a la que llama y las tablas de DynamoDB que usa para almacenar los datos. La instrumentación básica con filtros, complementos y clientes de

AWS SDK instrumentados se muestra en la rama del proyecto. `xray-gettingstarted` Esta es la ramificación que se implementa en el tutorial Introducción. Dado que esta ramificación solo incluye los aspectos básicos, puede diferenciarla rápidamente de la ramificación `master` para comprender rápidamente los aspectos básicos.



La aplicación de ejemplo muestra una instrumentación básica en estos archivos:

- Filtro de solicitudes HTTP: `WebConfig.java`
- AWS Instrumentación de cliente del SDK: `build.gradle`

La `xray` rama de la aplicación incluye el uso de anotaciones HTTPClient, consultas SQL, subsegmentos personalizados, una AWS Lambdafunción instrumentada y códigos y scripts de inicialización instrumentados.

Para permitir el inicio de sesión de los usuarios y AWS SDK para JavaScript su uso en el navegador, la `xray-cognito` sucursal añade Amazon Cognito para admitir la autenticación y la autorización de los usuarios. Con credenciales recuperadas desde Amazon Cognito, la aplicación web también envía datos de rastro a X-Ray para registrar la información de la solicitud desde el punto de vista del cliente. El cliente del navegador aparece como su propio nodo en el mapa de rastros y registra información adicional, incluida la URL de la página que el usuario está visualizando y el ID de usuario.

Por último, la ramificación `xray-worker` añade una función de Lambda en Python instrumentada que se ejecuta de forma independiente, procesando los elementos de una cola de Amazon SQS. Scorekeep añade un elemento a la cola cada vez que termina un juego. El trabajador de Lambda, activado por CloudWatch eventos, extrae los elementos de la cola cada pocos minutos y los procesa para almacenar los registros de los juegos en Amazon S3 para su análisis.

Temas

- Introducción a la aplicación de ejemplo Scorekeep
- Instrumentación AWS manual de los clientes del SDK
- Creación de subsegmentos adicionales
- Grabar anotaciones, metadatos y usuario IDs
- Instrumentación de llamadas a HTTP salientes
- Instrumentación de llamadas a una base de datos PostgreSQL
- Funciones de instrumentación AWS Lambda
- Instrumentación de código de inicio
- Instrumentación de scripts
- Instrumentación de un cliente de aplicación web
- Uso de clientes instrumentados en subprocesos de trabajo

Introducción a la aplicación de ejemplo Scorekeep

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

En este tutorial, se utiliza la `xray-gettingstarted` rama de la aplicación de ejemplo Scorekeep, que se utiliza CloudFormation para crear y configurar los recursos que ejecutan la aplicación de ejemplo y el daemon X-Ray en Amazon ECS. La aplicación utiliza el marco Spring para implementar una API web de JSON y AWS SDK for Java para conservar los datos en Amazon DynamoDB. Un servlet filtra en la aplicación instrumenta todas las solicitudes entrantes atendidas por la aplicación y un controlador de solicitudes en el cliente del AWS SDK instrumenta las llamadas descendentes a DynamoDB.

Puede seguir este tutorial utilizando el o el. Consola de administración de AWS AWS CLI

Secciones

Requisitos previos

Este tutorial se utiliza CloudFormation para crear y configurar los recursos que ejecutan la aplicación de ejemplo y el daemon X-Ray. Se tienen que cumplir los siguientes requisitos previos para instalar y ejecutar el tutorial:

1. Si utiliza un usuario de IAM con permisos limitados, añada las siguientes políticas de usuario en la consola de IAM:

   - `AWSCloudFormationFullAccess`— para acceder y usar CloudFormation
   - `AmazonS3FullAccess`— para cargar un archivo de plantilla CloudFormation utilizando el Consola de administración de AWS
   - `IAMFullAccess`— para crear los roles de EC2 instancia de Amazon ECS y Amazon
   - `AmazonEC2FullAccess`— crear los EC2 recursos de Amazon
   - `AmazonDynamoDBFullAccess`: para crear las tablas de DynamoDB
   - `AmazonECS_FullAccess`: para crear recursos de Amazon ECS
   - `AmazonSNSFullAccess`: para crear el tema de Amazon SNS
   - `AWSXrayReadOnlyAccess`: para obtener permiso para ver el mapa de rastros y los rastros en la consola de X-Ray
2. Para seguir el tutorial mediante el AWS CLI, instale la CLI versión 2.7.9 o posterior y configure la CLI con el usuario del paso anterior. Asegúrese de que la región esté configurada al configurar AWS CLI con el usuario. Si no se configura una región, hay que anexar `--region` *`AWS-REGION`* a todos los comandos de la CLI.
3. Asegúrese de que Git esté instalado para clonar el repositorio de aplicaciones de muestra.
4. Utilice el siguiente código de ejemplo para clonar la ramificación `xray-gettingstarted` del repositorio de Scorekeep:

```
git clone https://github.com/aws-samples/eb-java-scorekeep.git xray-scorekeep -b
 xray-gettingstarted
```

Instale la aplicación Scorekeep mediante CloudFormation

Consola de administración de AWS

Instale la aplicación de ejemplo mediante Consola de administración de AWS

1. Abra la consola de CloudFormation.
2. Elija Crear pila y, a continuación, elija Con nuevos recursos en el menú desplegable.
3. En la sección Especificar plantilla, elija Cargar un archivo de plantilla.
4. Seleccione Elegir archivo, navegue hasta la carpeta `xray-scorekeep/cloudformation` que se creó al clonar el repositorio de Git y elija el archivo `cf-resources.yaml`.
5. Elija Siguiente para continuar.
6. Escriba `scorekeep` en el cuadro de texto Nombre de la pila y, a continuación, seleccione Siguiente en la parte inferior de la página para continuar. Tenga en cuenta que en el resto de este tutorial se supone que la pila se llama `scorekeep`.
7. Desplázate hasta la parte inferior de la página Configuración de opciones de pila y seleccione Siguiente para continuar.
8. Desplázate hasta la parte inferior de la página de revisión, selecciona la casilla de verificación que indica que se CloudFormation pueden crear recursos de IAM con nombres personalizados y selecciona Crear pila.
9. La CloudFormation pila se está creando ahora. El estado de la pila será `CREATE_IN_PROGRESS` durante unos cinco minutos antes de cambiar a `CREATE_COMPLETE`. El estado se actualizará periódicamente, o el usuario puede actualizar la página.

AWS CLI

Instale la aplicación de ejemplo mediante AWS CLI

1. Navegue hasta la carpeta `cloudformation` del repositorio `xray-scorekeep` que clonó anteriormente en el tutorial:

```
cd xray-scorekeep/cloudformation/
```

2. Introduzca el siguiente AWS CLI comando para crear la CloudFormation pila:

```
aws cloudformation create-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

3. Espere hasta que el estado de la CloudFormation pila sea `CREATE_COMPLETE` correcto, lo que tardará unos cinco minutos. Usa el siguiente AWS CLI comando para comprobar el estado:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Generación de datos de rastreo

La aplicación de ejemplo incluye una aplicación web front-end. Utilice la aplicación web para generar tráfico a la API y enviar los datos de rastro a X-Ray. En primer lugar, recupere la URL de la aplicación web mediante la Consola de administración de AWS o la AWS CLI:

Consola de administración de AWS

Busque la URL de la aplicación mediante el Consola de administración de AWS

1. Abra la consola de CloudFormation.
2. Elija la pila `scorekeep` en la lista.
3. Elija la pestaña Salidas en la página de la pila `scorekeep` y elija el enlace URL `LoadBalancerUrl` para abrir la aplicación web.

AWS CLI

Busque la URL de la aplicación mediante el AWS CLI

1. Use el siguiente comando para mostrar la dirección URL de la aplicación web:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].Outputs[0].OutputValue"
```

2. Copie esta URL y ábrala en un navegador para mostrar la aplicación web Scorekeep.

Uso de la aplicación web para generar datos de rastro

1. Elija Create (Crear) para crear un usuario y una sesión.
2. Escriba un nombre de juego, establezca las Reglas en Tres en raya y elija después Crear para crear un juego.
3. Seleccione Play (Jugar) para comenzar el juego.
4. Elija una ficha para hacer un movimiento y cambiar el estado del juego.

Cada uno de estos pasos genera solicitudes HTTP a la API y llamadas posteriores a DynamoDB para leer y escribir el usuario, la sesión, el juego, la movida y los datos de estado.

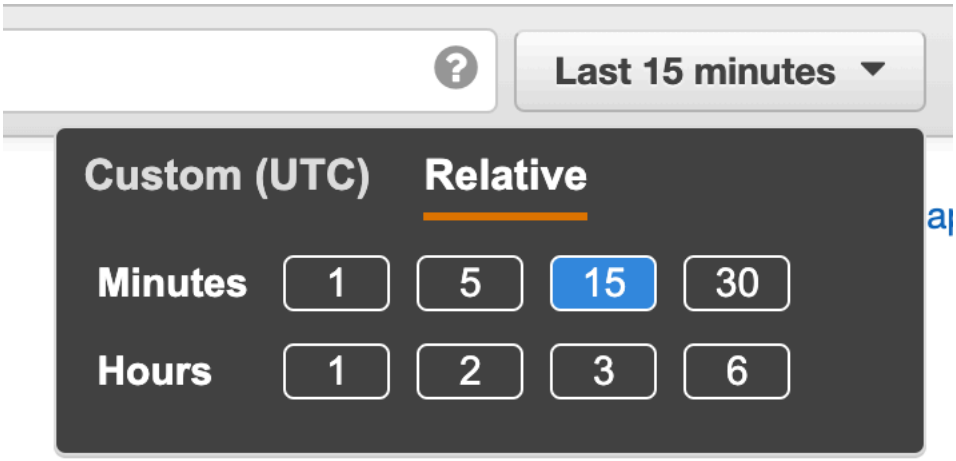
Vea el mapa de rastreo en el Consola de administración de AWS

Puede ver el mapa de trazas y las trazas generadas por la aplicación de ejemplo en X-Ray y en CloudWatch las consolas.

X-Ray console

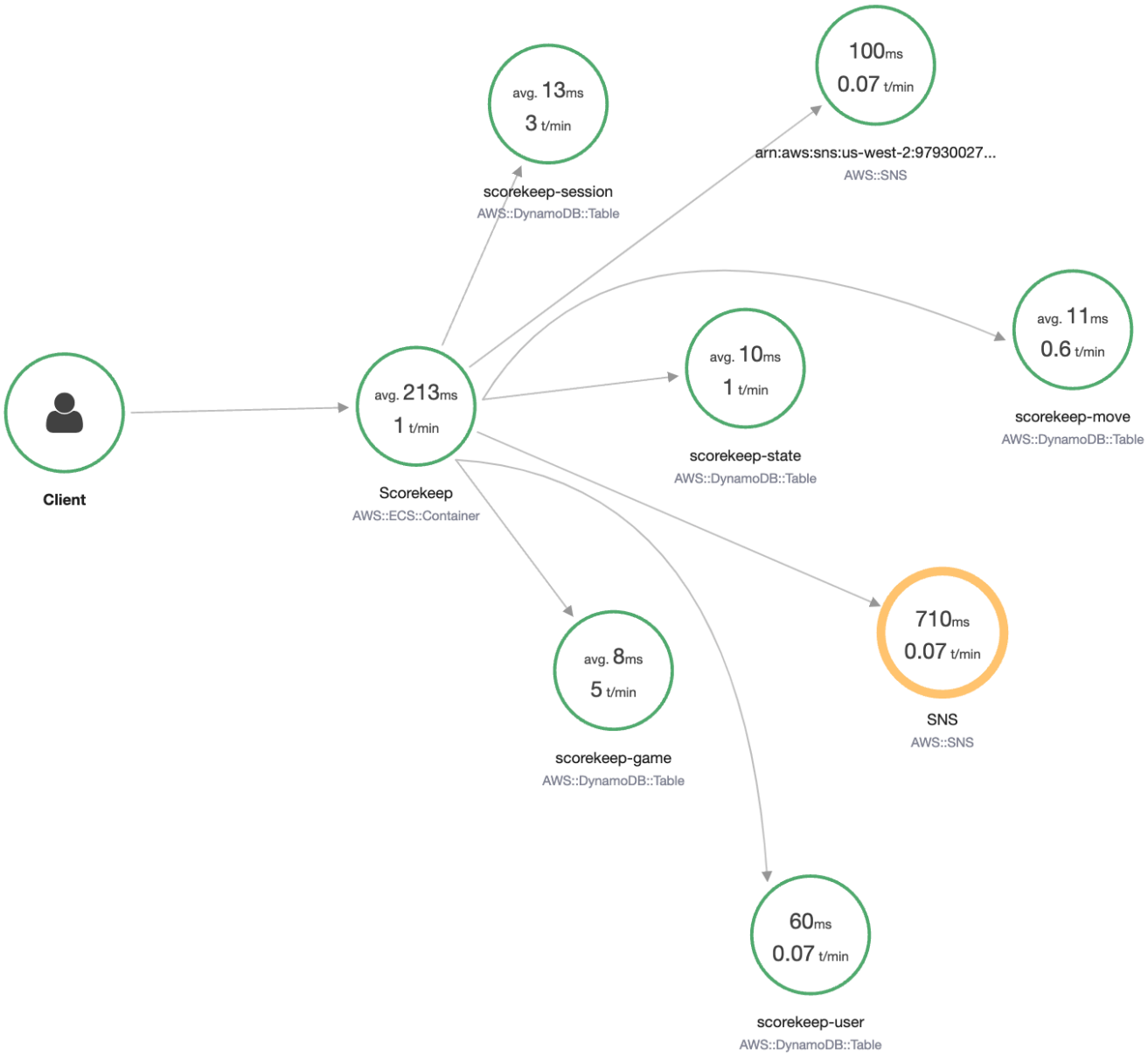
Uso de la consola de X-Ray

1. Abra la página del mapa de rastros de la consola de X-Ray.
2. La consola muestra una representación del gráfico de servicios que X-Ray genera a partir de los datos de rastro que envía la aplicación. Asegúrese de ajustar el período de tiempo del mapa de rastros si es necesario, para asegurarse de que mostrará todos los rastros desde que inició la aplicación web.



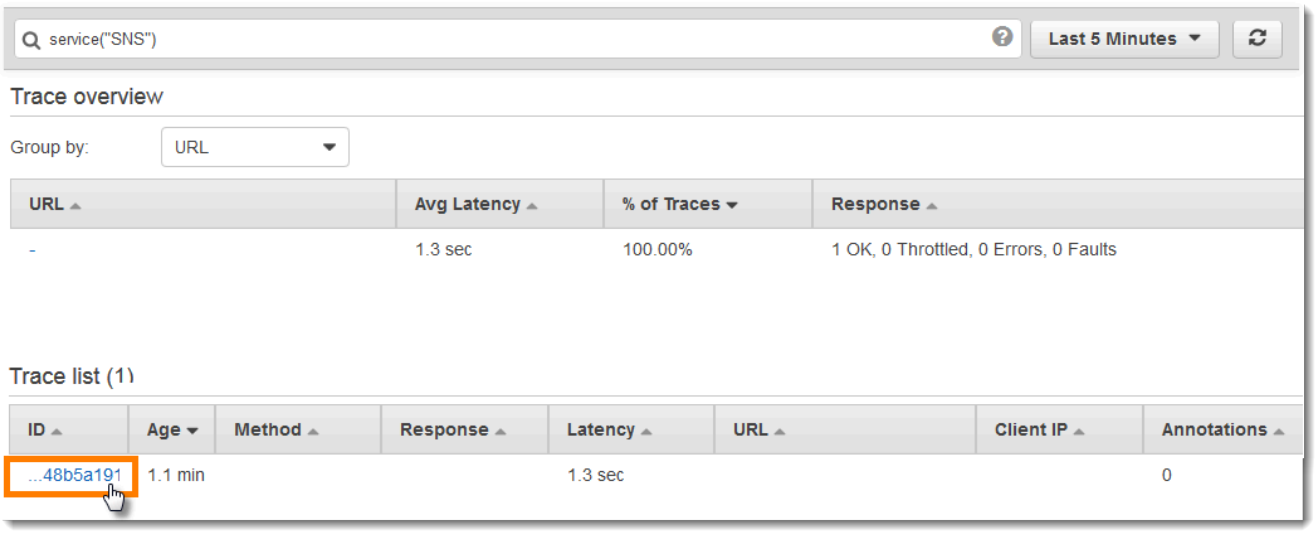
El mapa de rastros muestra el cliente de la aplicación web, la API que se ejecuta en Amazon ECS y cada tabla de DynamoDB que la aplicación utiliza. Cada solicitud que se envía a la aplicación, hasta una cantidad máxima configurable de solicitudes por segundo, se rastrea hasta que llega a la API, genera solicitudes para los servicios posteriores y se completa.

Puede elegir cualquier nodo en el gráfico de servicios para ver los rastros de las solicitudes que generaron tráfico hacia ese nodo. Actualmente, el nodo Amazon SNS es amarillo. Profundice para saber por qué.

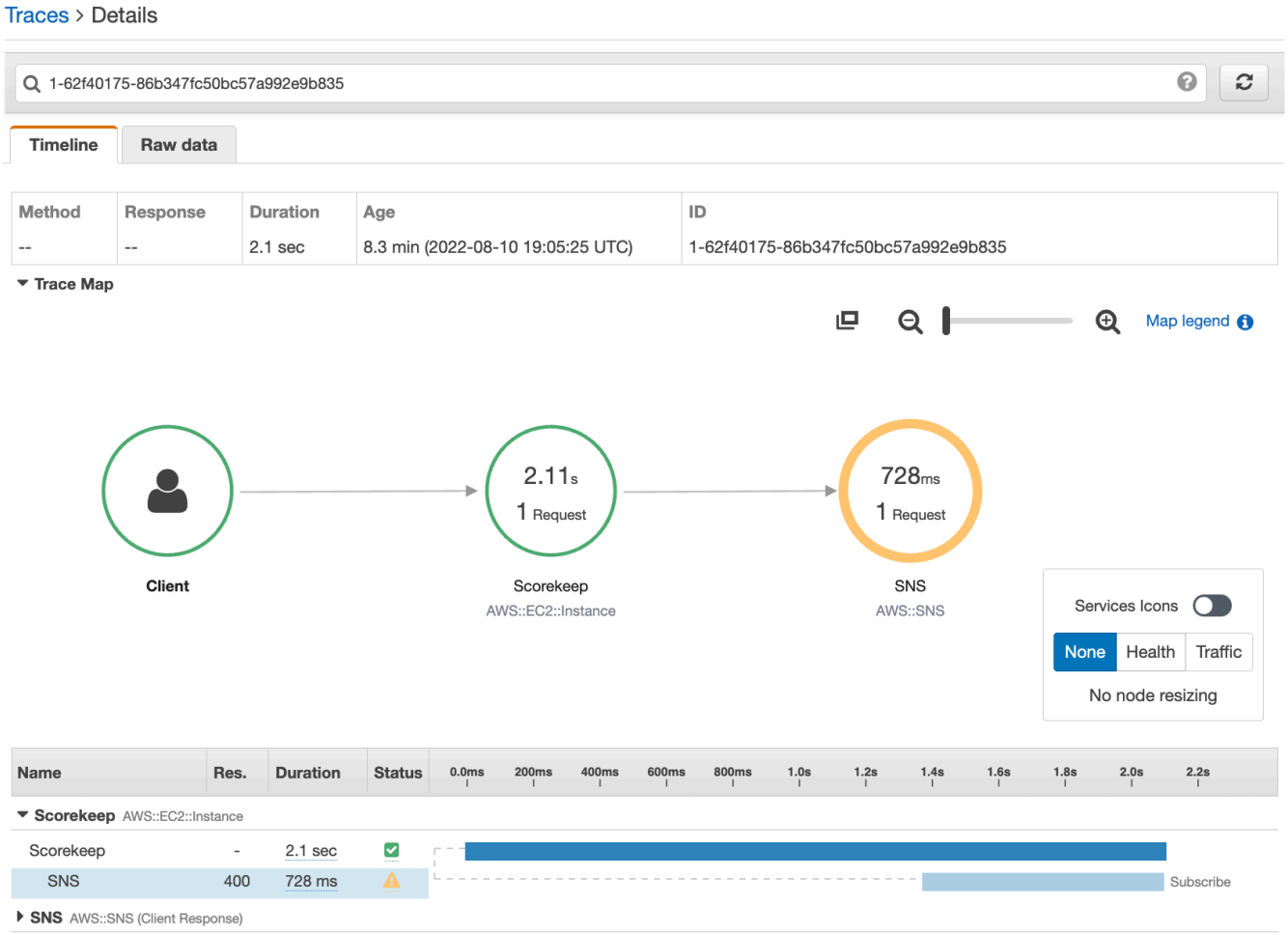


Para encontrar la causa del error

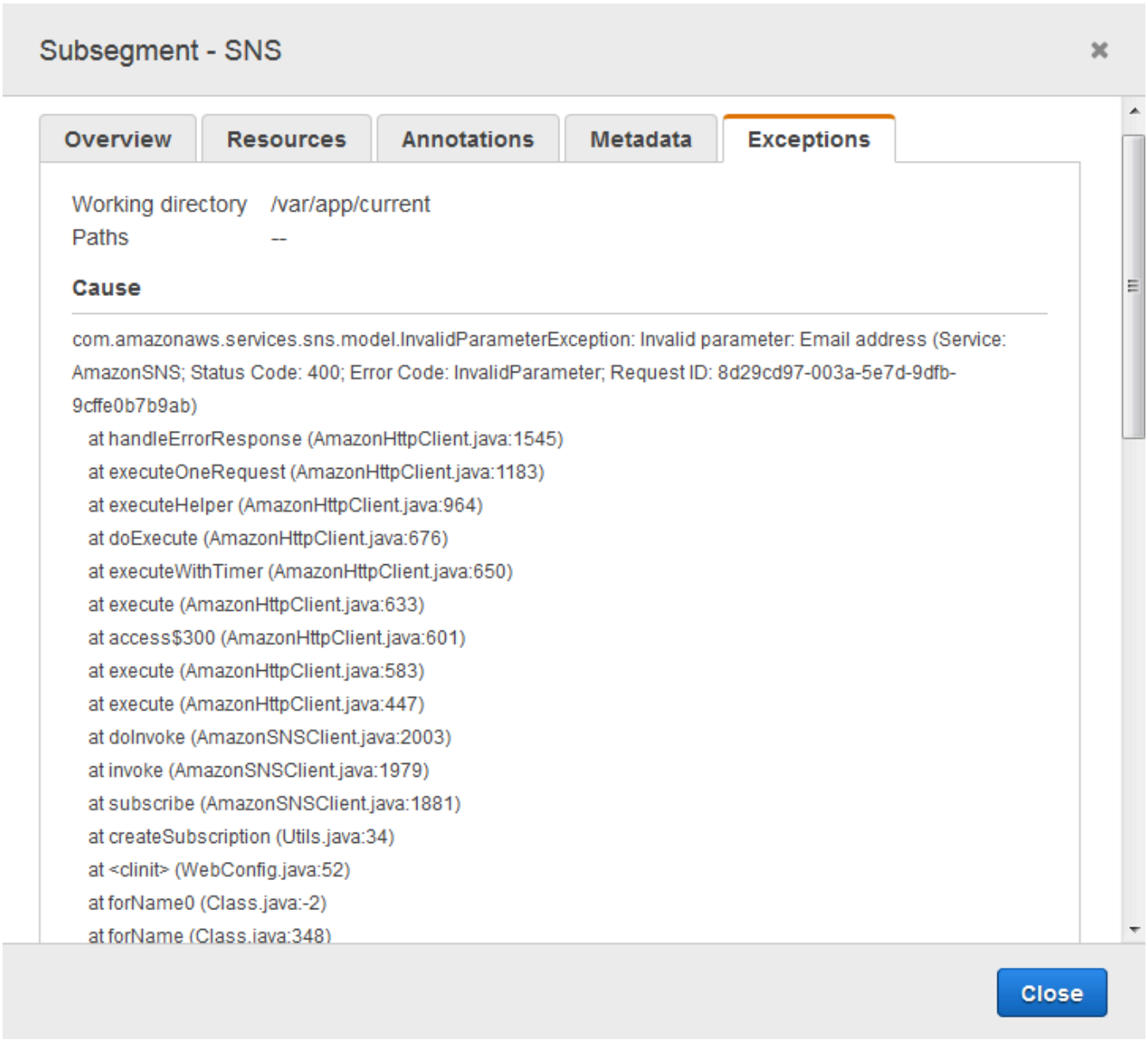
1. Seleccione el nodo denominado SNS. Se muestra el panel de detalles del nodo.
2. Elija View traces (Ver rastreos) para acceder a la pantalla Trace overview (Información general de rastreo).
3. Seleccione el rastro en Trace list (Lista de rastros). Este rastro no tiene método ni URL porque se registró durante el inicio y no en respuesta a una solicitud de entrada.



4. Seleccione el icono de estado de error en el segmento de Amazon SNS en la parte inferior de la página para abrir la página Excepciones del subsegmento de SNS.



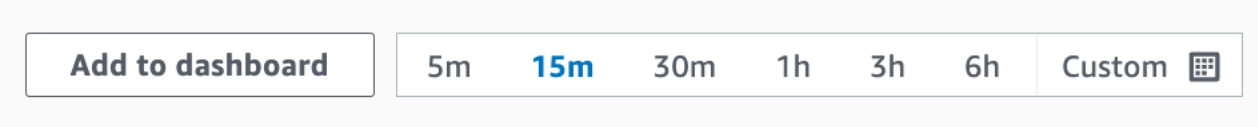
5. El SDK de X-Ray captura automáticamente las excepciones producidas por clientes del SDK de AWS instrumentados y registra el seguimiento de la pila.



CloudWatch console

Usa la CloudWatch consola

1. Abra la página del mapa de rastreo de X-Ray de la CloudWatch consola.
2. La consola muestra una representación del gráfico de servicios que X-Ray genera a partir de los datos de rastro que envía la aplicación. Asegúrese de ajustar el período de tiempo del mapa de rastros si es necesario, para asegurarse de que mostrará todos los rastros desde que inició la aplicación web.



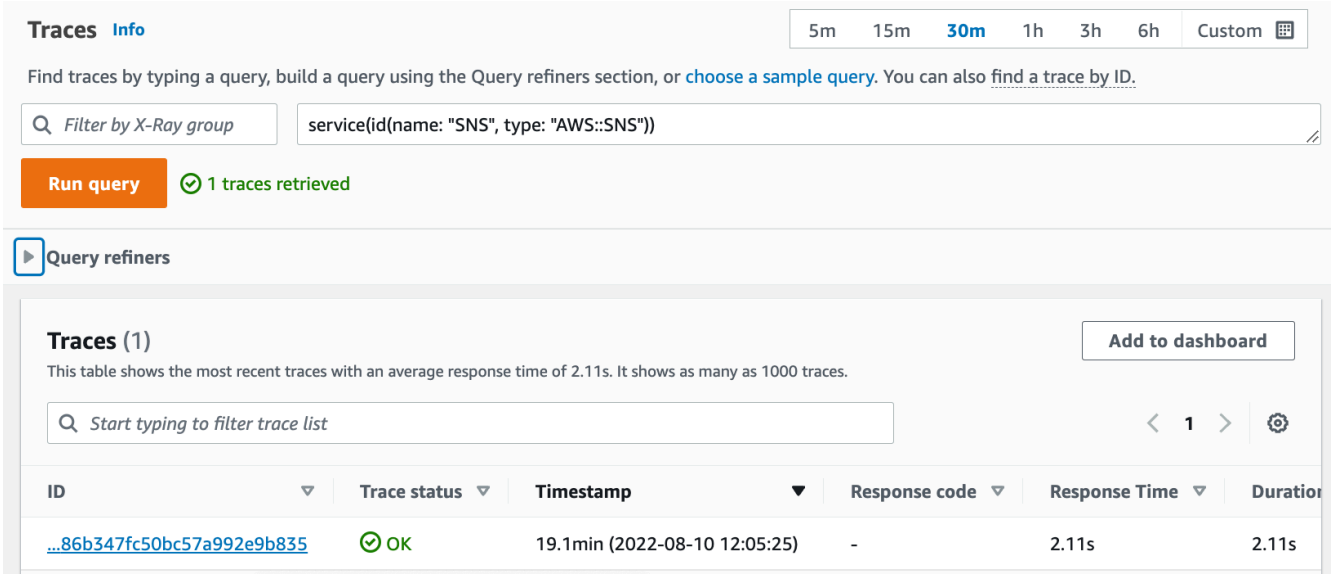
El mapa de seguimiento muestra el cliente de la aplicación web, la API que se ejecuta en Amazon EC2 y cada tabla de DynamoDB que utiliza la aplicación. Cada solicitud que se envía a la aplicación, hasta una cantidad máxima configurable de solicitudes por segundo, se rastrea hasta que llega a la API, genera solicitudes para los servicios posteriores y se completa.

Puede elegir cualquier nodo en el gráfico de servicios para ver los rastros de las solicitudes que generaron tráfico hacia ese nodo. Actualmente, el nodo Amazon SNS es naranja. Profundice para saber por qué.

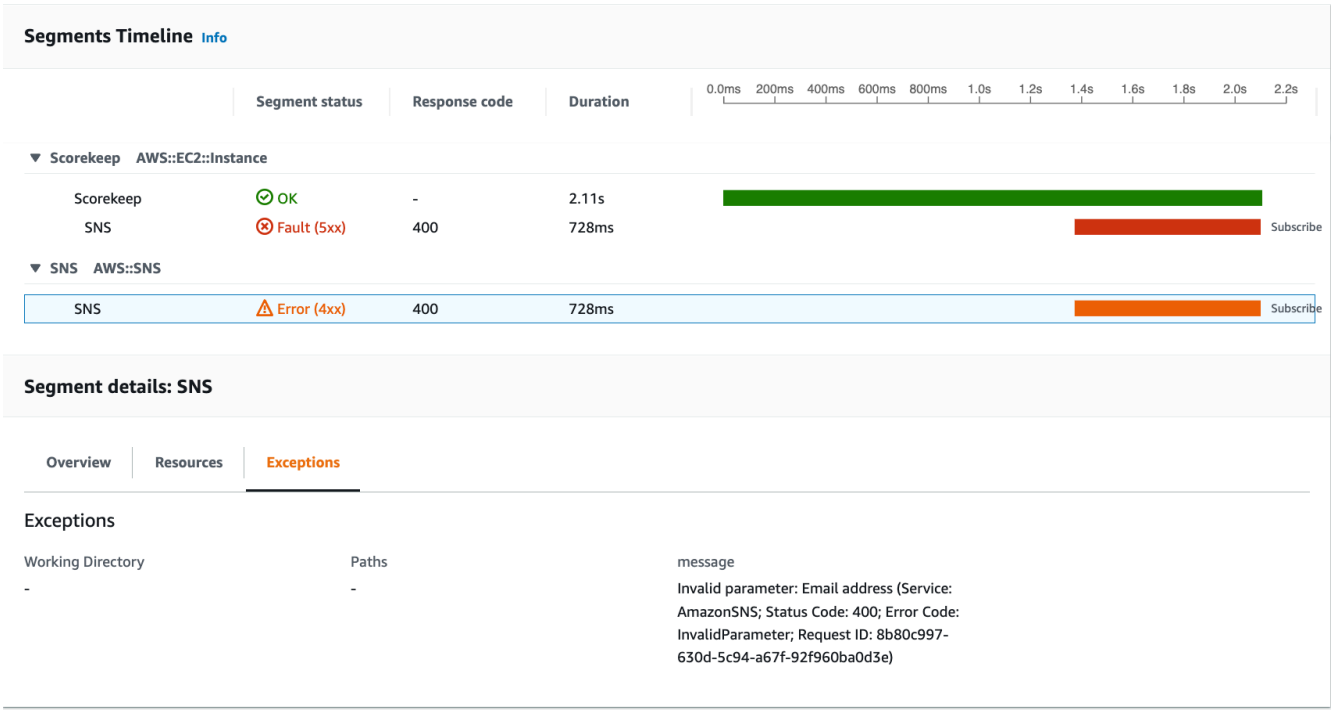

scorekeep-state
DynamoDB Table
scorekeep-game
DynamoDB Table
scorekeep-move
DynamoDB Table
Client
Scorekeep
ECS Container
SNS
SNS
scorekeep-session
DynamoDB Table

Para encontrar la causa del error

1. Seleccione el nodo denominado SNS. El panel de detalles del nodo SNS se muestra debajo del mapa.
2. Seleccione Ver rastros para acceder a la página Rastros.
3. En la parte inferior de la página, elija el rastro de la lista Rastros. Este rastro no tiene método ni URL porque se registró durante el inicio y no en respuesta a una solicitud de entrada.



4. Elija el subsegmento de Amazon SNS en la parte inferior de la escala de tiempo de los segmentos y elija la pestaña Excepciones del subsegmento de SNS para ver los detalles de la excepción.



La causa indica que la dirección de correo electrónico proporcionada en una llamada a `createSubscription` realizada en la clase `WebConfig` no era válida. Lo arreglaremos en la siguiente sección.

Configuración de notificaciones de Amazon SNS

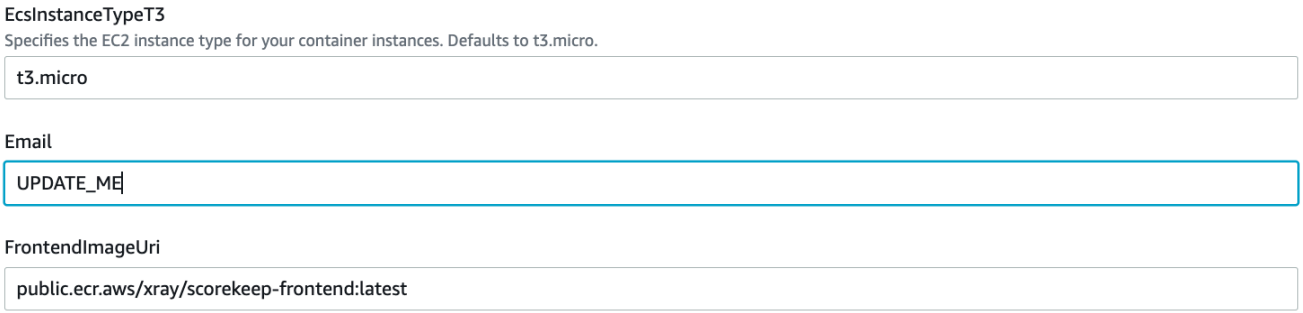
Scorekeep utiliza Amazon SNS para enviar notificaciones cuando los usuarios completan un juego. Cuando la aplicación se inicia, intenta crear una suscripción para una dirección de correo electrónico definida en un parámetro de CloudFormation pila. Esa llamada está fallando actualmente. Configure un correo electrónico de notificación para habilitar las notificaciones y resolver los errores destacados en el mapa de rastros.

Consola de administración de AWS

Para configurar las notificaciones de Amazon SNS mediante Consola de administración de AWS

1. Abra la consola de CloudFormation.
2. Seleccione el botón de opción situado junto al nombre de pila `scorekeep` en la lista y, a continuación, seleccione Actualizar.
3. Asegúrese de seleccionar Usar la plantilla actual y, a continuación, haga clic en Siguiente en la página Actualizar pila.

4. Busque el parámetro Correo electrónico en la lista y sustituya el valor predeterminado por una dirección de correo electrónico válida.



5. Desplácese hasta la parte inferior de la página y elija Siguiente.
6. Desplázate hasta la parte inferior de la página de revisión, selecciona la casilla de verificación que confirma que se CloudFormation pueden crear recursos de IAM con nombres personalizados y selecciona Actualizar pila.
7. La CloudFormation pila se está actualizando ahora. El estado de la pila será `UPDATE_IN_PROGRESS` durante unos cinco minutos antes de cambiar a `UPDATE_COMPLETE`. El estado se actualizará periódicamente, o el usuario puede actualizar la página.

AWS CLI

Para configurar las notificaciones de Amazon SNS mediante AWS CLI

1. Vaya a la carpeta `xray-scorekeep/cloudformation/` que creó anteriormente y abra el archivo `cf-resources.yaml` en un editor de texto.
2. Busque el `Default` valor en el parámetro Email y cámbielo por una dirección de *`UPDATE_ME`* correo electrónico válida.

```
Parameters:
  Email:
    Type: String
    Default: UPDATE_ME # <- change to a valid abc@def.xyz email address
```

3. Desde la `cloudformation` carpeta, actualiza la CloudFormation pila con el siguiente AWS CLI comando:

```
aws cloudformation update-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

4. Espera a que se CloudFormation muestre el estado de la pilaUPDATE_COMPLETE, lo que tardará unos minutos. Usa el siguiente AWS CLI comando para comprobar el estado:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Cuando se completa la actualización, Scorekeep se reinicia y crea una suscripción al tema de SNS. Compruebe su correo electrónico y confirme la suscripción para ver las actualizaciones cuando complete un juego. Abra el mapa de rastros para comprobar que las llamadas a SNS ya no están fallando.

Explorar la aplicación de ejemplo

La aplicación de ejemplo es una API web HTTP en Java que está configurada para utilizar el SDK de X-Ray para Java. Al implementar la aplicación con la CloudFormation plantilla, esta crea las tablas de DynamoDB, el clúster de Amazon ECS y otros servicios necesarios para ejecutar Scorekeep en ECS. Se crea un archivo de definición de tareas para ECS mediante. CloudFormation Este archivo define las imágenes del contenedor utilizadas por tarea en un clúster de ECS. Estas imágenes se obtienen del ECR público oficial de X-Ray. La imagen del contenedor de la API de Scorekeep tiene la API compilada con Gradle. La imagen de contenedor del contenedor frontend de Scorekeep atiende al frontend mediante el servidor proxy nginx. Este servidor dirige las solicitudes a las rutas que empiezan por /api a la API.

Para instrumentar las solicitudes de HTTP entrantes, la aplicación añade el `TracingFilter` que proporciona el SDK.

Example src/main/java/scorekeep/WebConfig.java: filtro de servlets

```
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
...

@Configuration
public class WebConfig {

  @Bean
```

```
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
...
```

Este filtro envía datos de rastreo sobre todas las solicitudes entrantes que la aplicación atiende, incluidos el URL, el método, el estado de la respuesta, la hora de inicio y la hora de finalización de la solicitud.

La aplicación también realiza llamadas posteriores a DynamoDB mediante el AWS SDK for Java. Para instrumentar estas llamadas, la aplicación simplemente toma los submódulos AWS relacionados con el SDK como dependencias y el X-Ray SDK for Java instrumenta automáticamente todos los clientes del SDK. AWS

La aplicación utiliza `Docker` para compilar el código de fuente de la instancia con `Gradle Docker Image` y el archivo `Scorekeep API Dockerfile` para ejecutar el archivo ejecutable JAR que Gradle genera en su `ENTRYPOINT`.

Example Uso de Docker para compilar a través de Gradle Docker Image

```
docker run --rm -v /PATH/TO/SCOREKEEP_REPO/home/gradle/project -w /home/gradle/project
 gradle:4.3 gradle build
```

Example ENTRYPOINT de Dockerfile

```
ENTRYPOINT [ "sh", "-c", "java -Dserver.port=5000 -jar scorekeep-api-1.0.0.jar" ]
```

El archivo `build.gradle` descarga todos los submódulos del SDK de Maven durante la compilación declarándolos como dependencias.

Example build.gradle: dependencias

```
...
dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    testCompile('org.springframework.boot:spring-boot-starter-test')
    compile('com.amazonaws:aws-java-sdk-dynamodb')
    compile("com.amazonaws:aws-xray-recorder-sdk-core")
    compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk")
    compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk-instrumentor")
    ...
```

```
}
dependencyManagement {
    imports {
        mavenBom("com.amazonaws:aws-java-sdk-bom:1.11.67")
        mavenBom("com.amazonaws:aws-xray-recorder-sdk-bom:2.11.0")
    }
}
```

Los submódulos core, AWS SDK y AWS SDK Instrumentor son todo lo que se necesita para instrumentar automáticamente cualquier llamada posterior realizada con el SDK. AWS

Para retransmitir datos de segmentos sin procesar a la API de X-Ray, el daemon de X-Ray debe escuchar el tráfico en el puerto UDP 2000. Para ello, la aplicación ejecuta el daemon de X-Ray en un contenedor que se implementa junto con la aplicación Scorekeep en ECS como contenedor asociado. Consulte el tema Daemon de X-Ray para obtener más información.

Example Definición del contenedor del daemon de X-Ray en una definición de tarea de ECS

```
...
Resources:
  ScorekeepTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      ...

      - Cpu: '256'
        Essential: true
        Image: amazon/aws-xray-daemon
        MemoryReservation: '128'
        Name: xray-daemon
        PortMappings:
          - ContainerPort: '2000'
            HostPort: '2000'
            Protocol: udp
      ...
```

El SDK de X-Ray proporciona una clase denominada `AWSXRay` que proporciona una grabadora global, un `TracingHandler` que se puede utilizar para instrumentar el código. Puede configurar la grabadora global para que personalice el `AWSXRayServletFilter` que crea los segmentos para las llamadas HTTP entrantes. La muestra incluye un bloque estático en la clase `WebConfig` que configura la grabadora global con complementos y reglas de muestreo.

Example src/main/java/scorekeep/WebConfig.java: grabadora

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.plugins.ECSPlugin;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;
...

@Configuration
public class WebConfig {
  ...

  static {
    AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
 ECSPlugin()).withPlugin(new EC2Plugin());

    URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
    builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

    AWSXRay.setGlobalRecorder(builder.build());
    ...

  }
}
```

En este ejemplo se utiliza el compilador para cargar las reglas de muestreo desde un archivo denominado `sampling-rules.json`. Las reglas de muestreo determinan la velocidad a la que el SDK registra los segmentos de las solicitudes entrantes.

Example src/main/java/resources/sampling-rules.json

```
{
  "version": 1,
  "rules": [
    {
      "description": "Resource creation.",
      "service_name": "*",
      "http_method": "POST",
      "url_path": "/api/*",
      "fixed_target": 1,
      "rate": 1.0
```

```
    },
    {
      "description": "Session polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/session/*",
      "fixed_target": 0,
      "rate": 0.05
    },
    {
      "description": "Game polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/game/*/*",
      "fixed_target": 0,
      "rate": 0.05
    },
    {
      "description": "State polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/state/*/*/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

El archivo de reglas de muestreo define cuatro reglas de muestreo personalizadas y la regla predeterminada. Para cada solicitud de entrada, el SDK evalúa las reglas personalizadas en el orden en que están definidas. El SDK aplica la primera regla que coincide con el método, la ruta y el nombre de servicio de la solicitud. Para Scorekeep, la primera regla captura todas las solicitudes POST (llamadas de creación de recursos) aplicando un objetivo fijo de una solicitud por segundo y una tasa del 1,0, es decir, el 100 % de las solicitudes después de cumplir el objetivo fijo.

Las otras tres reglas personalizadas aplican una tasa del 5 % sin objetivo fijo a las lecturas de sesiones, juegos y estado (solicitudes GET). De este modo se reduce al mínimo el número de rastros de las llamadas periódicas que el front-end realiza automáticamente cada pocos segundos para

asegurarse de que el contenido está actualizado. Para el resto de solicitudes, el archivo define una velocidad predeterminada de una solicitud por segundo y una tasa del 10 %.

La aplicación de ejemplo también muestra cómo usar funciones avanzadas, como la instrumentación manual de clientes SDK, la creación de subsegmentos adicionales y llamadas HTTP salientes. Para obtener más información, consulte AWS X-Ray ejemplo de aplicación.

Opcional: política de privilegios mínimos

Los contenedores ECS de Scorekeep acceden a los recursos mediante políticas de acceso total, como `AmazonSNSFullAccess` y `AmazonDynamoDBFullAccess`. El uso de políticas de acceso total no es la mejor práctica para las aplicaciones de producción. En el siguiente ejemplo se actualiza la política de IAM de DynamoDB para mejorar la seguridad de la aplicación. Para obtener más información sobre las mejores prácticas de seguridad en las políticas de IAM, consulte Administración de identidad y acceso para AWS X-Ray.

Example Definición del rol de la plantilla cf-resources.yaml ECSTask

```
ECSTaskRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
            Principal:
              Service:
                - "ecs-tasks.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      ManagedPolicyArns:
        - "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
        - "arn:aws:iam::aws:policy/AmazonSNSFullAccess"
        - "arn:aws:iam::aws:policy/AWSXrayFullAccess"
      RoleName: "scorekeepRole"
```

Para actualizar las políticas, primero identifique los ARN de sus recursos de DynamoDB. A continuación, utilice los ARN en una política de IAM personalizada. Por último, aplique esa política al perfil de instancia.

Para identificar el ARN de su recurso de DynamoDB:

1. Abra la consola de DynamoDB.
2. En la barra de navegación izquierda, elija Tablas.
3. Elija cualquiera de las opciones `scorekeep-*` para mostrar la página de detalles de la tabla.
4. En la pestaña Descripción general, seleccione Información adicional para expandir la sección y ver el nombre de recurso de Amazon (ARN). Copie este valor.
5. Inserte el ARN en la siguiente política de IAM y sustituya los valores `AWS_REGION` y `AWS_ACCOUNT_ID` por su región e ID de cuenta específicos. Esta nueva política solo permite las acciones especificadas, en lugar de la política `AmazonDynamoDBFullAccess`, que permite cualquier acción.

   Example

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ScorekeepDynamoDB",
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query"
            ],
            "Resource": "arn:aws:dynamodb:us-east-1:111122223333:table/
scorekeep-*"
        }
    ]
}
```

   Las tablas que crea la aplicación siguen una convención de nomenclatura coherente. Puede usar el formato `scorekeep-*` para indicar todas las tablas de Scorekeep.

Cambio de la política de IAM

1. Abra el rol de tarea de Scorekeep (ScorekeepRole) desde la consola de IAM.
2. Elija la casilla de verificación situada junto a la política `AmazonDynamoDBFullAccess` y, a continuación, elija Eliminar para eliminar esta política.
3. Seleccione Añadir permisos, a continuación, Adjuntar políticas y, por último, Crear política.
4. Elija la pestaña JSON y pegue la política que acaba de crear.
5. Elija Siguiente: Etiquetas en la parte inferior de la página.
6. Elija Siguiente: Revisar en la parte inferior de la página.
7. En Nombre, escriba un nombre para la política.
8. Elija Crear política en la parte inferior de la página.
9. Adjunte la política que acaba de crear al rol `scorekeepRole`. Puede que la política adjunta tarde unos minutos en aplicarse.

Si ha adjuntado la nueva política al `scorekeepRole` rol, debe separarla antes de eliminar la CloudFormation pila, ya que esta política adjunta impedirá que se elimine la pila. La política se puede separar automáticamente eliminándola.

Eliminación de su política de IAM personalizada

1. Abra la consola de IAM.
2. Elija Políticas en la barra de navegación izquierda.
3. Busque el nombre de la política personalizada que creó anteriormente en esta sección y elija el botón de opción situado junto al nombre de la política para resaltarlo.
4. Abra la lista desplegable Acciones y, a continuación, elija Eliminar.
5. Escriba el nombre de la política personalizada y, a continuación, elija Eliminar para confirmar la eliminación. Esto separará automáticamente la política del rol `scorekeepRole`.

Limpieza

Siga estos pasos para eliminar los recursos de la aplicación Scorekeep:

> **Note**
>
> Si creó y adjuntó políticas personalizadas mediante la sección anterior de este tutorial, debe eliminar la política de la pila `scorekeepRole` antes de eliminar la CloudFormation pila.

Consola de administración de AWS

Elimine la aplicación de muestra mediante el Consola de administración de AWS

1. Abra la consola de CloudFormation.
2. Seleccione el botón de opción situado junto al nombre de pila `scorekeep` en la lista y, a continuación, seleccione Eliminar.
3. La CloudFormation pila se está borrando ahora. El estado de la pila será `DELETE_IN_PROGRESS` de unos minutos hasta que se eliminen todos los recursos. El estado se actualizará periódicamente, o el usuario puede actualizar la página.

AWS CLI

Elimine la aplicación de muestra mediante el AWS CLI

1. Introduzca el siguiente AWS CLI comando para eliminar la CloudFormation pila:

```
aws cloudformation delete-stack --stack-name scorekeep
```

2. Espere hasta que la CloudFormation pila deje de existir, lo que tardará unos cinco minutos. Usa el siguiente AWS CLI comando para comprobar el estado:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Siguientes pasos

Obtenga más información sobre X-Ray en el próximo capítulo: AWS X-RayConceptos de.

Para crear tu propia aplicación, obtén más información sobre el X-Ray SDK para Java o uno de los otros X-Ray SDKs:

- SDK de X-Ray para Java: AWS X-Ray SDK for Java
- SDK de X-Ray para Node.js: AWS SDK de X-Ray para Node.js
- SDK de X-Ray para .NET: AWS X-Ray SDK para.NET

Para ejecutar el daemon de X-Ray de forma local o activada AWS, consulteAWS X-Ray demonio.

Para contribuir al ejemplo de la aplicación GitHub, consulte eb-java-scorekeep.

Instrumentación AWS manual de los clientes del SDK

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El X-Ray SDK para Java instrumenta automáticamente todos los clientes del AWS SDK al incluir el submódulo AWS SDK Instrumentor en las dependencias de compilación.

Puede deshabilitar la instrumentación de clientes automática eliminando el submódulo Instrumentor. Esto le permite instrumentar algunos clientes manualmente a la vez que se pasan otros por alto, o utilizar diferentes controladores de rastreo en clientes distintos.

Para ilustrar la compatibilidad con la instrumentación de clientes de AWS SDK específicos, la aplicación pasa un controlador de rastreo `AmazonDynamoDBClientBuilder` como controlador de solicitudes en el modelo de usuario, juego y sesión. Esta modificación del código indica al SDK que instrumente todas las llamadas a DynamoDB con esos clientes.

Example **`src/main/java/scorekeep/SessionModel.java`**:instrumentación manual de clientes del SDK de AWS

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.handlers.TracingHandler;
```

```
public class SessionModel {
  private AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
        .withRegion(Constants.REGION)
        .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
        .build();
  private DynamoDBMapper mapper = new DynamoDBMapper(client);
```

Si eliminas el submódulo AWS SDK Instrumentor de las dependencias del proyecto, solo los clientes del SDK AWS instrumentados manualmente aparecen en el mapa de seguimiento.

Creación de subsegmentos adicionales

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

En la clase de modelo del usuario, la aplicación crea de forma manual subsegmentos para agrupar todas las llamadas posteriores que se realizaron dentro de la función `saveUser` y agrega metadatos.

Example **`src/main/java/scorekeep/UserModel.java`**: subsegmentos personalizados

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Subsegment;
...
    public void saveUser(User user) {
    // Wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## UserModel.saveUser");
    try {
      mapper.save(user);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
```

```
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

Grabar anotaciones, metadatos y usuario IDs

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

En la clase de modelo de juego, la aplicación registra objetos `Game` en un bloque de metadatos cada vez que se guarda un juego en DynamoDB. Por separado, la aplicación graba el juego IDs en anotaciones para usarlo con las expresiones de filtro.

Example **`src/main/java/scorekeep/GameModel.java`**: anotaciones y metadatos

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## GameModel.saveGame");
    try {
      // check session
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
        throw new SessionNotFoundException(sessionId);
      }
      Segment segment = AWSXRay.getCurrentSegment();
      subsegment.putMetadata("resources", "game", game);
      segment.putAnnotation("gameid", game.getId());
```

```
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

En el controlador de movimiento, la aplicación graba al usuario IDs `consetUser`. IDsLos usuarios se registran en un campo separado en los segmentos y se indexan para usarlos en la búsqueda.

Example src/main/java/scorekeep/MoveController.java: ID de usuario

```
import com.amazonaws.xray.AWSXRay;
...
  @RequestMapping(value="/{userId}", method=RequestMethod.POST)
  public Move newMove(@PathVariable String sessionId, @PathVariable String
 gameId, @PathVariable String userId, @RequestBody String move) throws
 SessionNotFoundException, GameNotFoundException, StateNotFoundException,
 RulesException {
    AWSXRay.getCurrentSegment().setUser(userId);
    return moveFactory.newMove(sessionId, gameId, userId, move);
  }
```

Instrumentación de llamadas a HTTP salientes

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

La clase del generador del usuario muestra cómo la aplicación usa el SDK de X-Ray para la versión de Java de `HTTPClientBuilder` con el fin de instrumentar las llamadas HTTP salientes.

Example **src/main/java/scorekeep/UserFactory.java**— instrumentación HTTPClient

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;

  public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
    HttpGet httpGet = new HttpGet("http://uinames.com/api/");
    CloseableHttpResponse response = httpclient.execute(httpGet);
    try {
      HttpEntity entity = response.getEntity();
      InputStream inputStream = entity.getContent();
      ObjectMapper mapper = new ObjectMapper();
      Map<String, String> jsonMap = mapper.readValue(inputStream, Map.class);
      String name = jsonMap.get("name");
      EntityUtils.consume(entity);
      return name;
    } finally {
      response.close();
    }
  }
```

Si actualmente utiliza `org.apache.http.impl.client.HttpClientBuilder`, puede simplemente intercambiar la instrucción de importación para esa clase por una para `com.amazonaws.xray.proxies.apache.http.HttpClientBuilder`.

Instrumentación de llamadas a una base de datos PostgreSQL

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

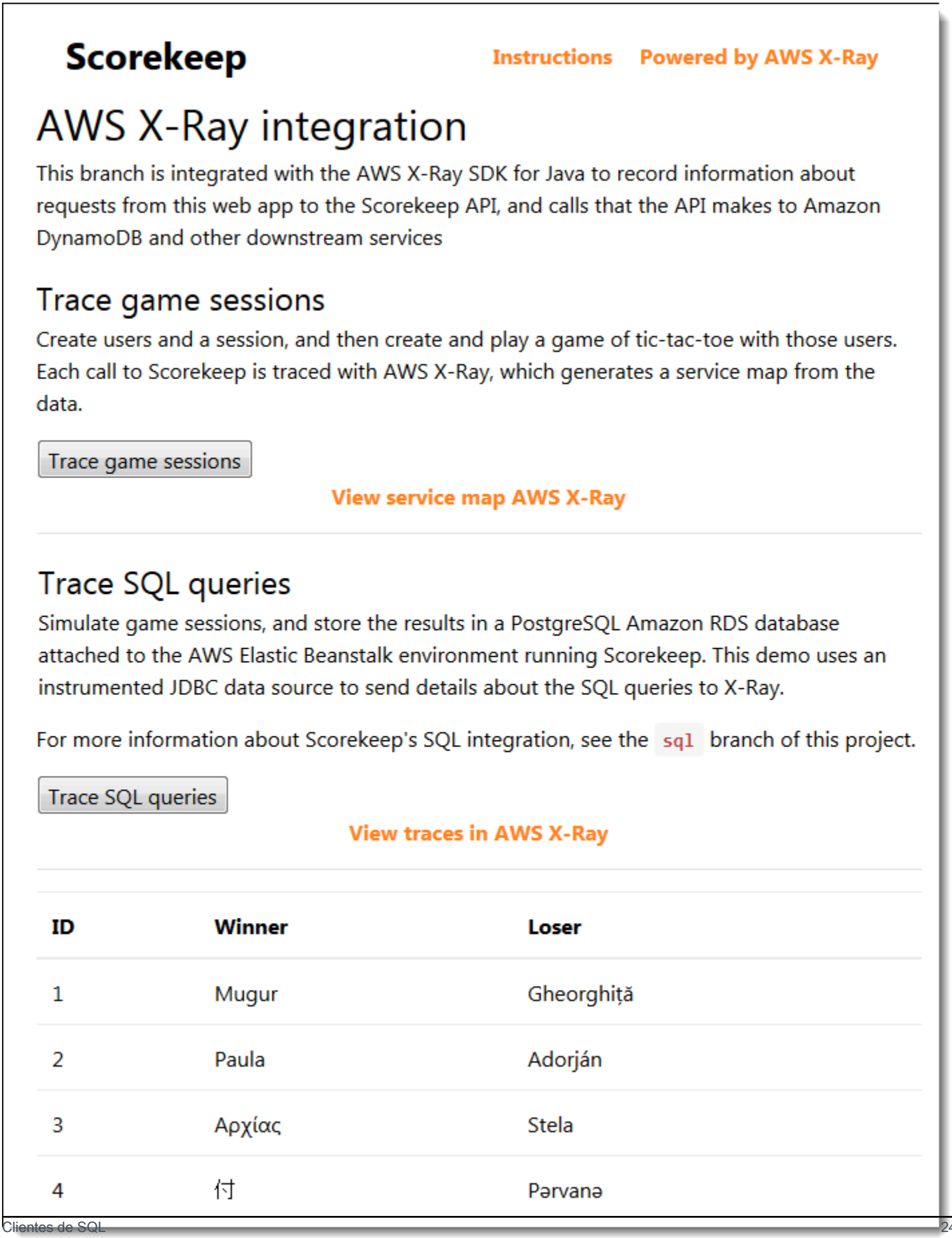
El archivo `application-pgsql.properties` añade el interceptor de rastreo PostgreSQL de X-Ray al origen de datos creado en `RdsWebConfig.java`.

Example **`application-pgsql.properties`**: instrumentación de una base de datos PostgreSQL

```
spring.datasource.continue-on-error=true
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.jdbc-interceptors=com.amazonaws.xray.sql.postgres.TracingInterceptor
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

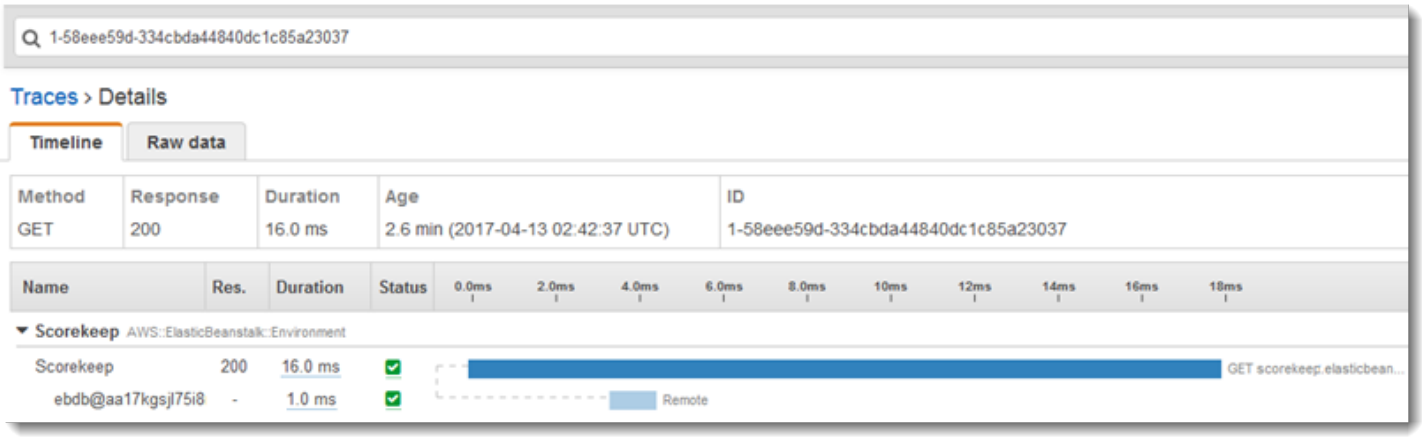
> **Note**
>
> Consulte Configuración de bases de datos con Elastic Beanstalk en la Guía para desarrolladores de AWS Elastic Beanstalk para obtener más información acerca de cómo añadir una base de datos de PostgreSQL al entorno de aplicaciones.

La página de demostración de X-Ray en la ramificación `xray` incluye una demostración que utiliza el origen de datos instrumentado para generar rastros que muestran información sobre las consultas SQL que se generan. Vaya a la ruta `/#/xray` de la aplicación en ejecución o elija Powered by AWS X-Ray en la barra de navegación para ver la página de demostración.


Scorekeep
Instructions
Powered by AWS X-Ray
AWS X-Ray integration
This branch is integrated with the AWS X-Ray SDK for Java to record information about requests from this web app to the Scorekeep API, and calls that the API makes to Amazon DynamoDB and other downstream services
Trace game sessions
Create users and a session, and then create and play a game of tic-tac-toe with those users. Each call to Scorekeep is traced with AWS X-Ray, which generates a service map from the data.
Trace game sessions
View service map AWS X-Ray
Trace SQL queries
Simulate game sessions, and store the results in a PostgreSQL Amazon RDS database attached to the AWS Elastic Beanstalk environment running Scorekeep. This demo uses an instrumented JDBC data source to send details about the SQL queries to X-Ray.
For more information about Scorekeep's SQL integration, see the sql branch of this project.
Trace SQL queries
View traces in AWS X-Ray
ID
Winner
Loser
1
Mugur
Gheorghiță
2
Paula
Adorján
3
Αρχίας
Stela
4
付
Pərvanə

Elija Trace SQL queries (Rastrear consultas SQL) para simular las sesiones de juego y almacenar los resultados en la base de datos asociada. A continuación, selecciona Ver trazas en AWS X-Ray para ver una lista filtrada de las trazas que llegan a la `/api/history` ruta de la API.

Elija uno de los rastros de la lista para ver la escala de tiempo, incluida la consulta SQL.



Funciones de instrumentación AWS Lambda

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Scorekeep utiliza dos funciones. AWS Lambda La primera es una función de Node.js de la ramificación `lambda` que genera nombres aleatorios para nuevos usuarios. Cuando un usuario crea una sesión sin introducir ningún nombre, la aplicación llama a una función denominada `random-name` con el AWS SDK for Java. El SDK de X-Ray para Java registra la información sobre la llamada a Lambda en un subsegmento, como cualquier otra llamada realizada con un cliente de SDK AWS instrumentado.

> **Note**
>
> La ejecución de la función de Lambda `random-name` requiere la creación de recursos adicionales fuera del entorno de Elastic Beanstalk. Consulte el archivo readme para obtener más información e instrucciones: AWS Lambda Integration.

La segunda función, `scorekeep-worker`, es una función de Python que se ejecuta de forma independiente de la API de Scorekeep. Cuando un juego termina, la API escribe el ID de sesión y el ID de juego en una cola de SQS. La función de proceso de trabajo lee los elementos de la cola y llama a la API de Scorekeep con el fin de construir registros completos de cada sesión de juego para su almacenamiento en Amazon S3.

Scorekeep incluye CloudFormation plantillas y scripts para crear ambas funciones. Dado que necesita agrupar el SDK de X-Ray con el código de la función, las plantillas crean las funciones sin ningún tipo de código. Al implementar Scorekeep, un archivo de configuración incluido en la carpeta `.ebextensions` crea un paquete de origen que incluye el SDK y actualiza el código de la función y la configuración con el AWS Command Line Interface.

Funciones

- Nombre aleatorio
- Entorno de trabajo

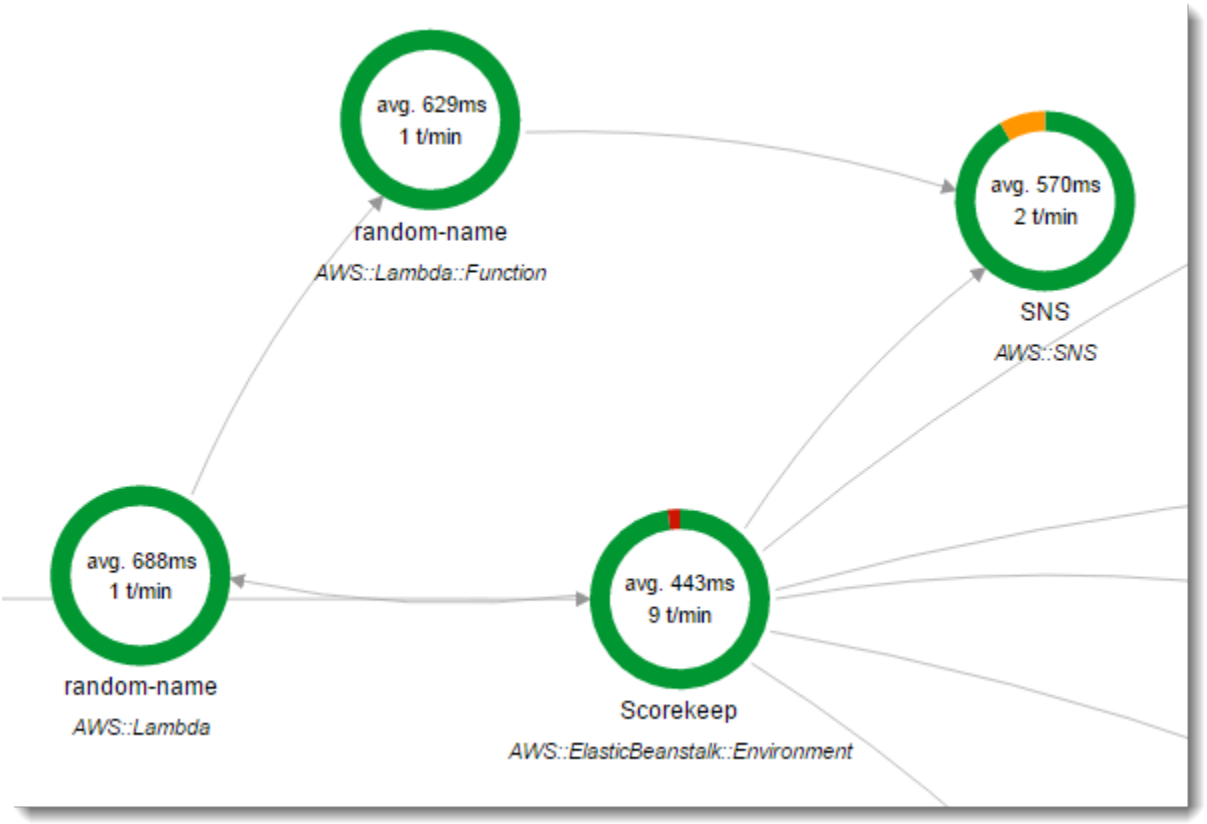
Nombre aleatorio

Scorekeep llama a la función de nombre aleatorio cuando un usuario comienza una sesión de juego sin iniciar sesión o especificar un nombre de usuario. Cuando Lambda procesa la llamada a `random-name`, lee el encabezado de rastreo, que contiene el ID de rastro y la decisión de muestreo escrita por el SDK de X-Ray para Java.

Para cada solicitud muestreada, Lambda ejecuta el daemon de X-Ray y escribe dos segmentos. El primer segmento registra información sobre la llamada a Lambda que invoca la función. Este segmento contiene la misma información que el subsegmento registrado por Scorekeep, pero desde el punto de vista de Lambda. El segundo segmento representa el trabajo que hace la función.

Lambda pasa el segmento de la función al SDK de X-Ray a través del contexto de la función. Cuando instrumente una función de Lambda, no utilice el SDK con el fin de crear un segmento

para las solicitudes entrantes. Lambda proporciona el segmento y usted debe utilizar el SDK para instrumentar clientes y escribir subsegmentos.



La función `random-name` se implementa en Node.js. Utiliza el SDK de Node.js para JavaScript enviar notificaciones con Amazon SNS y el SDK de X-Ray para Node.js para instrumentar el cliente del AWS SDK. Para escribir anotaciones, la función crea un subsegmento personalizado con `AWSXRay.captureFunc` y escribe anotaciones en la función instrumentada. En Lambda, no puede escribir anotaciones directamente en el segmento de la función, únicamente en un subsegmento que cree.

Example **`function/index.js`**: función Lambda random-name

```
var AWSXRay = require('aws-xray-sdk-core');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));

AWS.config.update({region: process.env.AWS_REGION});
var Chance = require('chance');

var myFunction = function(event, context, callback) {
  var sns = new AWS.SNS();
  var chance = new Chance();
```

```
  var userid = event.userid;
  var name = chance.first();

  AWSXRay.captureFunc('annotations', function(subsegment){
    subsegment.addAnnotation('Name', name);
    subsegment.addAnnotation('UserID', event.userid);
  });

  // Notify
  var params = {
    Message: 'Created randon name "' + name + '"" for user "' + userid + '".',
    Subject: 'New user: ' + name,
    TopicArn: process.env.TOPIC_ARN
  };
  sns.publish(params, function(err, data) {
    if (err) {
      console.log(err, err.stack);
      callback(err);
    }
    else {
      console.log(data);
      callback(null, {"name": name});
    }
  });
};

exports.handler = myFunction;
```

Esta función se crea automáticamente al implementar la aplicación de ejemplo en Elastic Beanstalk. La ramificación `xray` incluye un script para crear una función de Lambda en blanco. Los archivos de configuración de la `.ebextensions` carpeta crean el paquete de funciones `npm install` durante la implementación y, a continuación, actualizan la función Lambda con la CLI AWS .

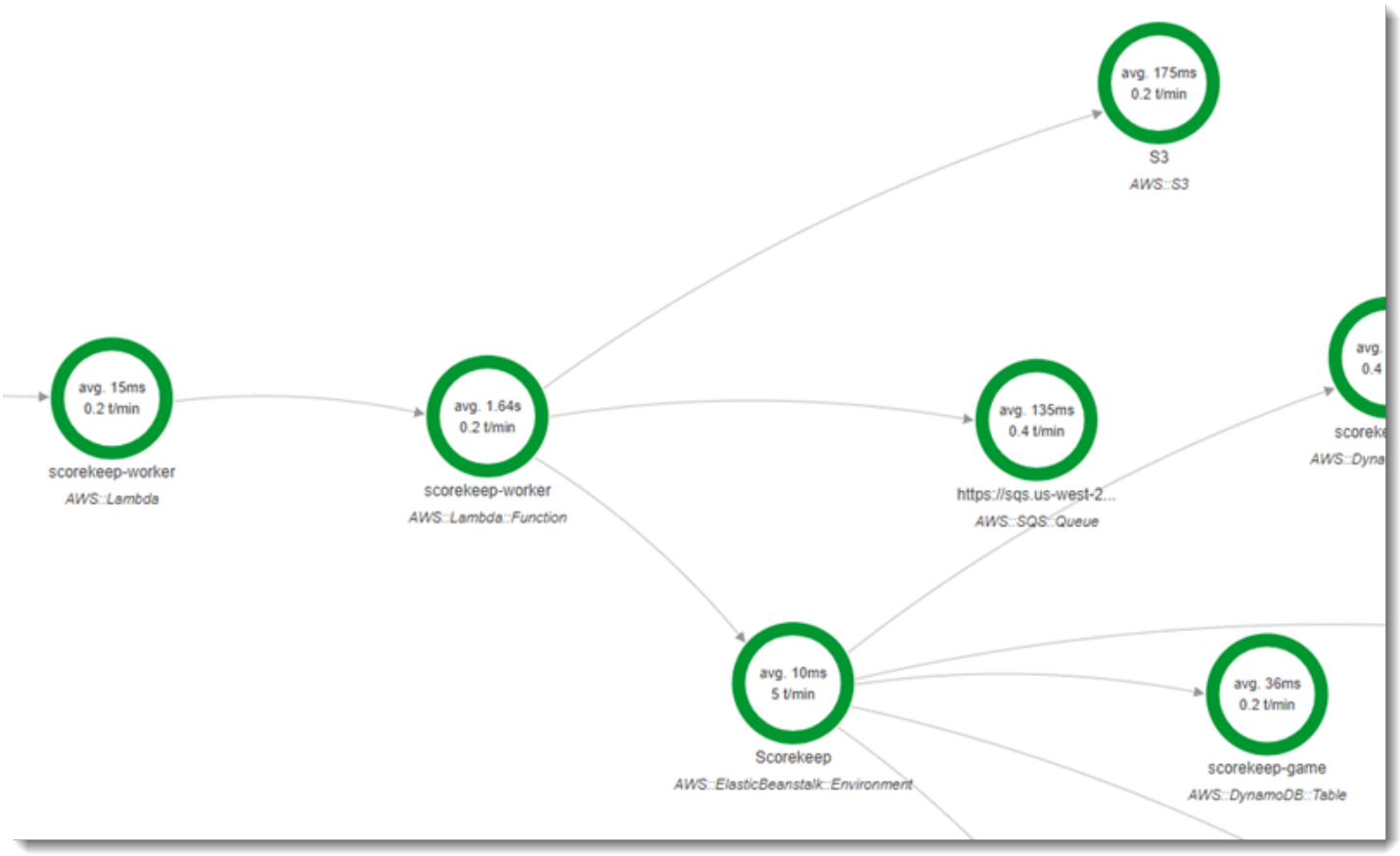
Entorno de trabajo

La función de trabajo instrumentada se proporciona en su propia ramificación, `xray-worker`, ya que no se puede ejecutar a menos que antes cree la función de trabajo y los recursos relacionados. Consulte el archivo readme de la ramificación para obtener instrucciones.

La función se activa mediante un paquete de CloudWatch eventos de Amazon Events cada 5 minutos. Cuando se ejecuta, la función extrae un elemento de una cola de Amazon SQS que administra Scorekeep. Cada mensaje contiene información sobre un juego terminado.

El trabajo extrae los documentos y el registro del juego de otras tablas a las que hace referencia el registro de juego. Por ejemplo, el registro del juego en DynamoDB incluye una lista de los movimientos que se han realizado durante el juego. La lista no contiene los movimientos propiamente dichos, sino IDs los movimientos que están guardados en una tabla aparte.

Las sesiones y los estados también se almacenan como referencias. Esto impide que las entradas de la tabla de juego sean demasiado grandes, pero requiere llamadas adicionales para obtener toda la información sobre el juego. El proceso de trabajo anula las referencias a todas estas entradas y construye un registro completo del juego como un único documento en Amazon S3. Cuando desee realizar el análisis de los datos, podrá ejecutar consultas en él directamente en Amazon S3 con Amazon Athena sin ejecutar migraciones de datos que realizan muchas lecturas para obtener los datos de DynamoDB.



La función de trabajo tiene habilitado el rastreo activo en su configuración en AWS Lambda. A diferencia de la función de nombres aleatorios, el trabajador no recibe una solicitud de una aplicación instrumentada, por lo que AWS Lambda no recibe un encabezado de rastreo. Con el rastreo activo, Lambda crea el ID de rastro y toma decisiones de muestreo.

El X-Ray SDK para Python está solo unas líneas en la parte superior de la función que importa el SDK y ejecuta su `patch_all` función para parchear el AWS SDK para Python (Boto) y HTTclients que utiliza para llamar a Amazon SQS y Amazon S3. Cuando el trabajo llama a la API de Scorekeep, el SDK añade el encabezado de rastreo a la solicitud para rastrear llamadas a través de la API.

Example_lambda/scorekeep-worker/scorekeep-worker.py: función Lambda del proceso de trabajo

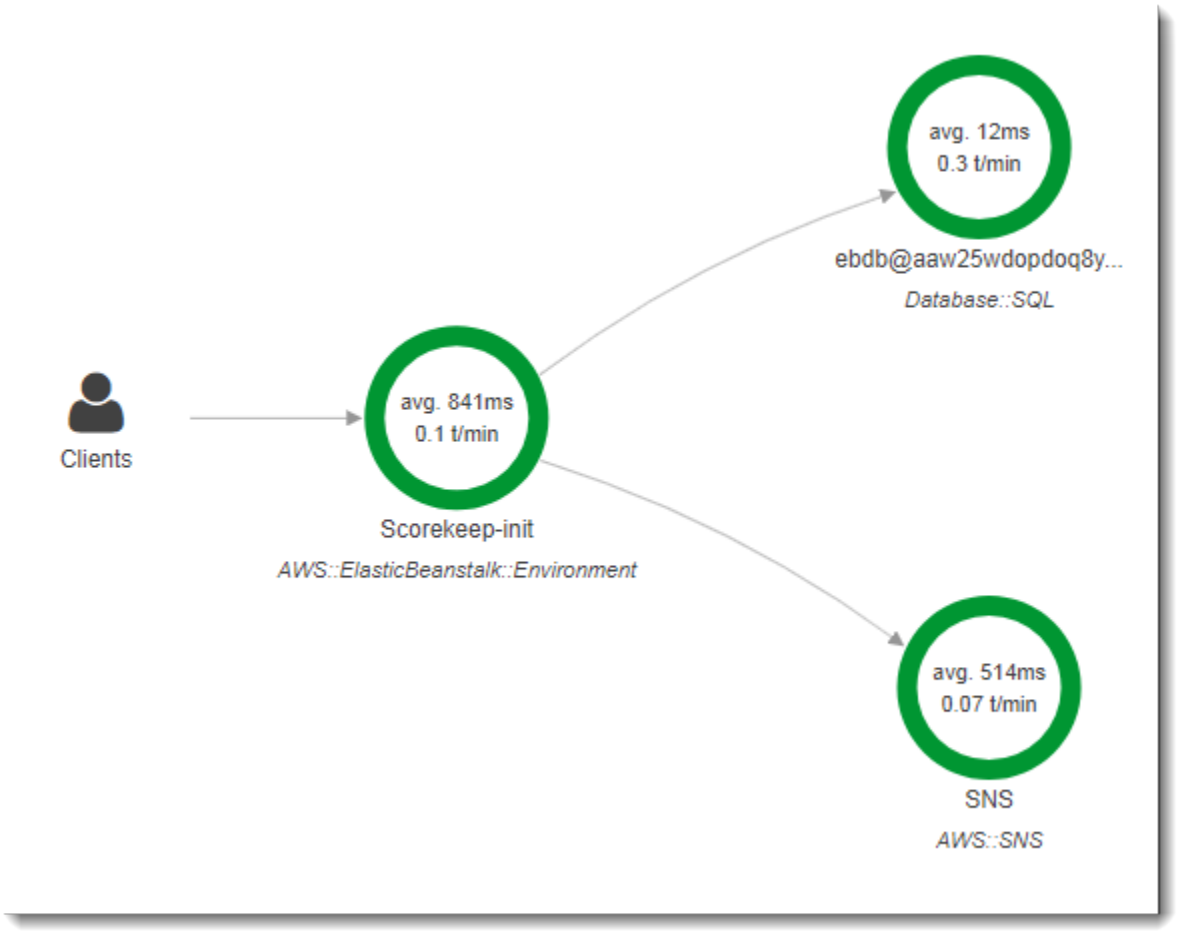
```
import os
import boto3
import json
import requests
import time
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

patch_all()
queue_url = os.environ['WORKER_QUEUE']

def lambda_handler(event, context):
    # Create SQS client
    sqs = boto3.client('sqs')
    s3client = boto3.client('s3')

    # Receive message from SQS queue
    response = sqs.receive_message(
        QueueUrl=queue_url,
        AttributeNames=[
            'SentTimestamp'
        ],
        MaxNumberOfMessages=1,
        MessageAttributeNames=[
            'All'
        ],
        VisibilityTimeout=0,
        WaitTimeSeconds=0
    )
   ...
```

Instrumentación de código de inicio

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Java crea automáticamente segmentos para solicitudes entrantes. Siempre que haya una solicitud en el ámbito, puede utilizar clientes instrumentados y registrar subsegmentos sin problema. Sin embargo, si intentas utilizar un cliente instrumentado en el código de inicio, obtendrás un. SegmentNotFoundException

El código de inicio se ejecuta fuera del request/response flujo estándar de una aplicación web, por lo que es necesario crear segmentos manualmente para instrumentalizarlo. Scorekeep muestra la instrumentación de código de inicio en sus archivos `WebConfig`. Scorekeep llama a una base de datos SQL y Amazon SNS durante el startup.



La clase `WebConfig` predeterminada crea una suscripción de Amazon SNS para notificaciones. Para proporcionar un segmento para que el SDK de X-Ray escriba cuando se utiliza el cliente de Amazon SNS, Scorekeep llama a `beginSegment` y `endSegment` en la grabadora global.

Example **src/main/java/scorekeep/WebConfig.java**: cliente del SDK de AWS instrumentado en el código de inicio

```
AWSXRay.beginSegment("Scorekeep-init");
if ( System.getenv("NOTIFICATION_EMAIL") != null ){
  try { Sns.createSubscription(); }
  catch (Exception e ) {
    logger.warn("Failed to create subscription for email "+
 System.getenv("NOTIFICATION_EMAIL"));
  }
}
AWSXRay.endSegment();
```

En `RdsWebConfig`, que Scorekeep utiliza cuando hay una base de datos de Amazon RDS conectada, la configuración también crea un segmento para el cliente SQL que utiliza Hibernate cuando aplica el esquema de base de datos durante el startup.

Example **`src/main/java/scorekeep/RdsWebConfig.java`**: cliente de base de datos SQL instrumentado en código de inicio

```
@PostConstruct
public void schemaExport() {
  EntityManagerFactoryImpl entityManagerFactoryImpl = (EntityManagerFactoryImpl)
 localContainerEntityManagerFactoryBean.getNativeEntityManagerFactory();
  SessionFactoryImplementor sessionFactoryImplementor =
 entityManagerFactoryImpl.getSessionFactory();
  StandardServiceRegistry standardServiceRegistry =
 sessionFactoryImplementor.getSessionFactoryOptions().getServiceRegistry();
  MetadataSources metadataSources = new MetadataSources(new
 BootstrapServiceRegistryBuilder().build());
  metadataSources.addAnnotatedClass(GameHistory.class);
  MetadataImplementor metadataImplementor = (MetadataImplementor)
 metadataSources.buildMetadata(standardServiceRegistry);
  SchemaExport schemaExport = new SchemaExport(standardServiceRegistry,
 metadataImplementor);

  AWSXRay.beginSegment("Scorekeep-init");
  schemaExport.create(true, true);
  AWSXRay.endSegment();
}
```

`SchemaExport` se ejecuta de forma automática y utiliza un cliente SQL. Dado que el cliente está instrumentado, Scorekeep debe anular la implementación predeterminada y proporcionar un segmento para que lo utilice el SDK cuando se llama al cliente.

Instrumentación de scripts

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y

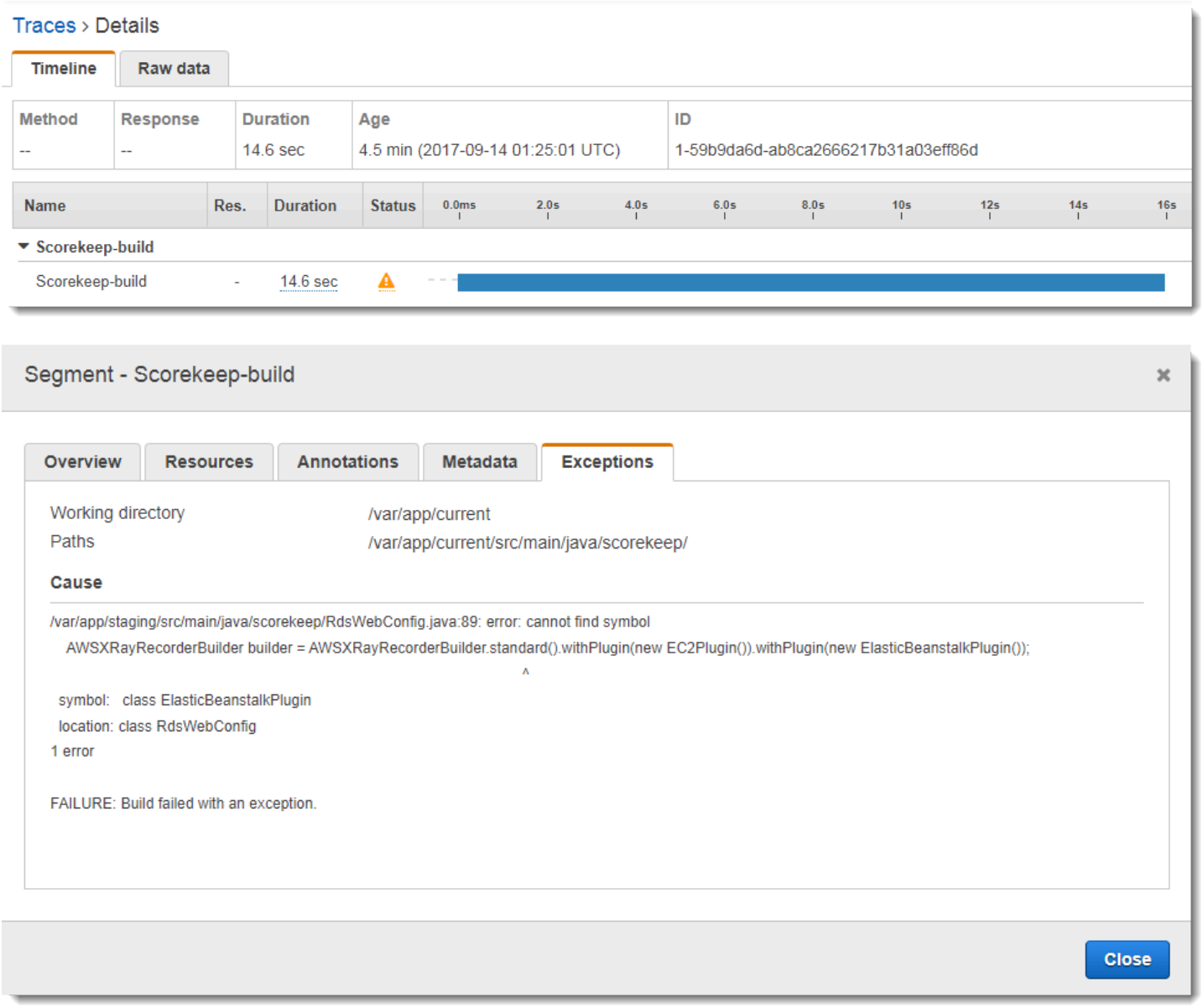
Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

También puede instrumentar código que no forme parte de su aplicación. Cuando el daemon de X-Ray se está ejecutando, transmitirá los segmentos que reciba a X-Ray, incluso si no los ha generado el SDK de X-Ray. Scorekeep utiliza sus propios scripts para instrumentar la compilación que compila la aplicación durante la implementación.

Example **`bin/build.sh`**: script de compilación instrumentado

```
SEGMENT=$(python bin/xray_start.py)
gradle build --quiet --stacktrace &> /var/log/gradle.log; GRADLE_RETURN=$?
if (( GRADLE_RETURN != 0 )); then
  echo "Gradle failed with exit status $GRADLE_RETURN" >&2
  python bin/xray_error.py "$SEGMENT" "$(cat /var/log/gradle.log)"
  exit 1
fi
python bin/xray_success.py "$SEGMENT"
```

`xray_start.py`, `xray_error.py` y `xray_success.py` son scripts de Python sencillos que construyen objetos de segmento, los convierten a documentos JSON y los envían al demonio sobre UDP. Si la compilación Gradle falla, puede encontrar el mensaje de error haciendo clic en el nodo scorekeep-build en el mapa de rastros de la consola de X-Ray.



Instrumentación de un cliente de aplicación web

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información

sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

En la ramificación `xray-cognito`, Scorekeep utiliza Amazon Cognito para permitir que los usuarios creen una cuenta e inicien sesión con ella para recuperar su información de usuario desde un grupo de usuarios de Amazon Cognito. Cuando un usuario inicia sesión, Scorekeep utiliza un grupo de identidades de Amazon Cognito para obtener credenciales AWS temporales para usarlas con. AWS SDK para JavaScript

El grupo de identidades está configurado para permitir a los usuarios que han iniciado sesión escribir datos de rastreos en AWS X-Ray. La aplicación web utiliza estas credenciales para registrar el ID del usuario que ha iniciado sesión, la ruta del navegador y la vista del cliente de llamadas a la API Scorekeep.

La mayor parte del trabajo se realiza en una clase de servicio denominada `xray`. Este clase de servicio ofrece métodos para generar los identificadores requeridos, crear segmentos en curso, finalizar segmentos y enviar documentos de segmento a la API de X-Ray.

Example **`public/xray.js`**: registrar y cargar segmentos

```
...
  service.beginSegment = function() {
    var segment = {};
    var traceId = '1-' + service.getHexTime() + '-' + service.getHexId(24);

    var id = service.getHexId(16);
    var startTime = service.getEpochTime();

    segment.trace_id = traceId;
    segment.id = id;
    segment.start_time = startTime;
    segment.name = 'Scorekeep-client';
    segment.in_progress = true;
    segment.user =  sessionStorage['userid'];
    segment.http = {
      request: {
        url: window.location.href
      }
    };

    var documents = [];
```

```
    documents[0] = JSON.stringify(segment);
    service.putDocuments(documents);
    return segment;
  }

  service.endSegment = function(segment) {
    var endTime = service.getEpochTime();
    segment.end_time = endTime;
    segment.in_progress = false;
    var documents = [];
    documents[0] = JSON.stringify(segment);
    service.putDocuments(documents);
  }

  service.putDocuments = function(documents) {
    var xray = new AWS.XRay();
    var params = {
      TraceSegmentDocuments: documents
    };
    xray.putTraceSegments(params, function(err, data) {
      if (err) {
        console.log(err, err.stack);
      } else {
        console.log(data);
      }
    })
  }
```

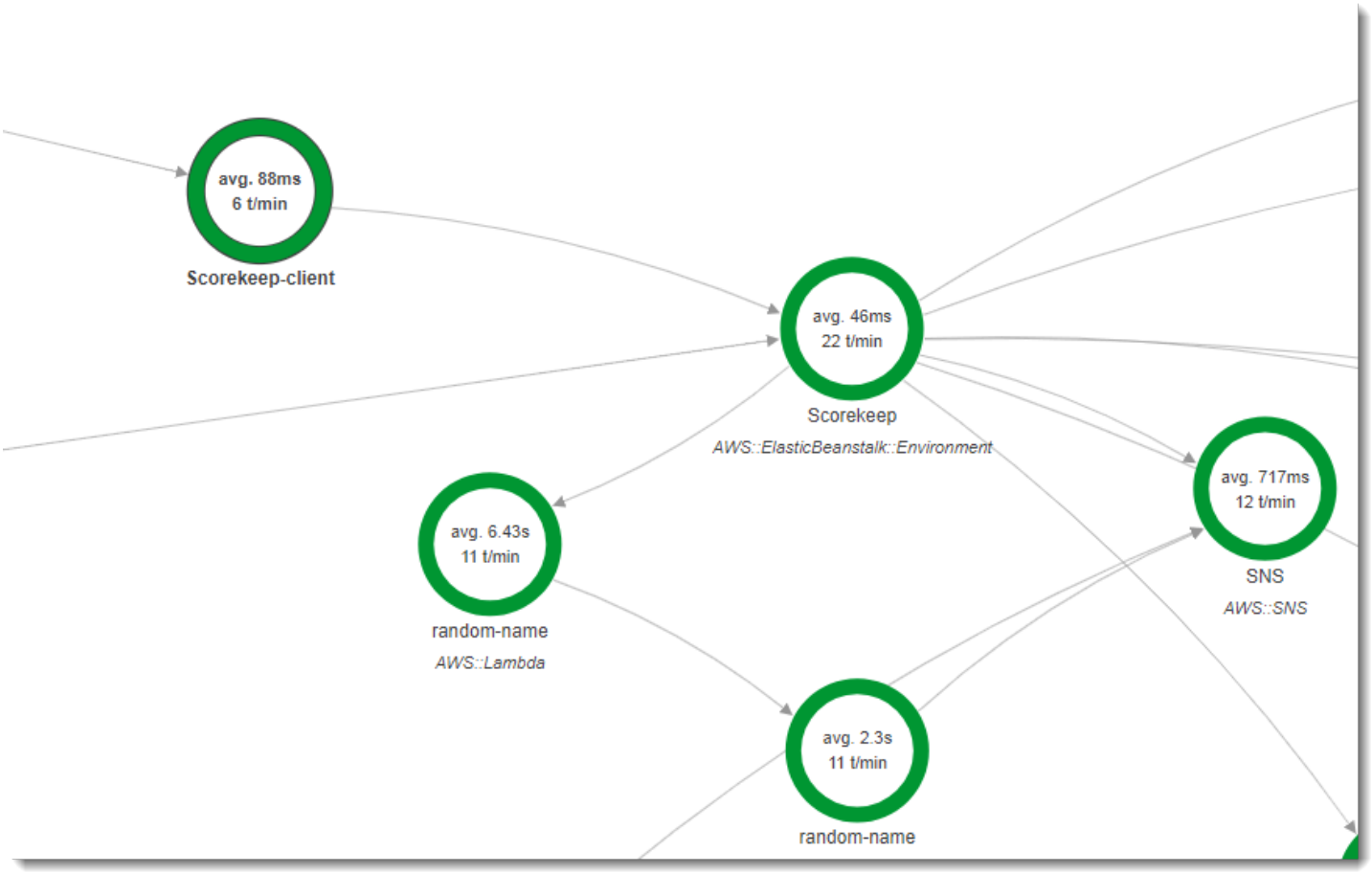
Estos métodos se llaman en el encabezado y funciones `transformResponse` en los servicios de recursos que la aplicación web utiliza para llamar a la API de Scorekeep. Para incluir el segmento de cliente en el mismo rastreo que el segmento que genera la API, la aplicación web debe incluir el ID de rastro y el ID de segmento en un encabezado de seguimiento (`X-Amzn-Trace-Id`) que el SDK de X-Ray puede leer. Cuando la aplicación Java instrumentada recibe una solicitud con este encabezado, el SDK de X-Ray para Java utiliza el mismo ID de rastro y hace que el segmento del cliente de la aplicación web sea el principal de su segmento.

Example **`public/app/services.js`**: registro de segmentos para llamadas de recursos de Angular y escritura de encabezados de rastreo

```
var module = angular.module('scorekeep');
module.factory('SessionService', function($resource, api, XRay) {
  return $resource(api + 'session/:id', { id: '@_id' }, {
```

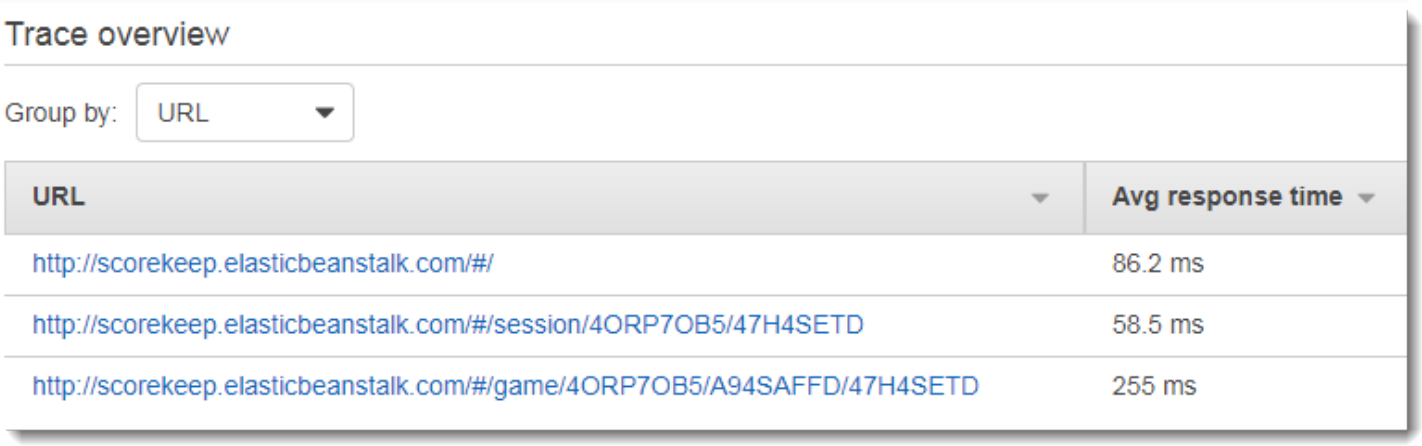
```
    segment: {},
    get: {
      method: 'GET',
      headers: {
        'X-Amzn-Trace-Id': function(config) {
          segment = XRay.beginSegment();
          return XRay.getTraceHeader(segment);
        }
      },
      transformResponse: function(data) {
        XRay.endSegment(segment);
        return angular.fromJson(data);
      },
    },
...
```

El mapa de rastros resultante incluye un nodo para el cliente de la aplicación web.



Los rastreos que incluyen segmentos desde la aplicación web muestran la dirección URL que el usuario ve en el navegador (rutas que empiezan por `/#/`). Sin la instrumentación de cliente, solo

obtiene la dirección URL del recurso de la API que la aplicación web llama (rutas que empiezan por `/api/`).



Uso de clientes instrumentados en subprocesos de trabajo

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Scorekeep utiliza un subproceso de trabajo para publicar una notificación en Amazon SNS cuando un usuario gana un juego. La publicación de la notificación tarda más tiempo que el resto de las operaciones de solicitud combinadas y no afecta al cliente o al usuario. Por lo tanto, la realización de la tarea de forma asíncrona es una buena manera de mejorar el tiempo de respuesta.

Sin embargo, el SDK de X-Ray para Java no sabe qué segmento estaba activo cuando se creó el subproceso. Como resultado, cuando intentas utilizar el AWS SDK for Java cliente instrumentado dentro del hilo, se lanza un `SegmentNotFoundException` hilo y se bloquea el hilo.

Example Web-1.error.log

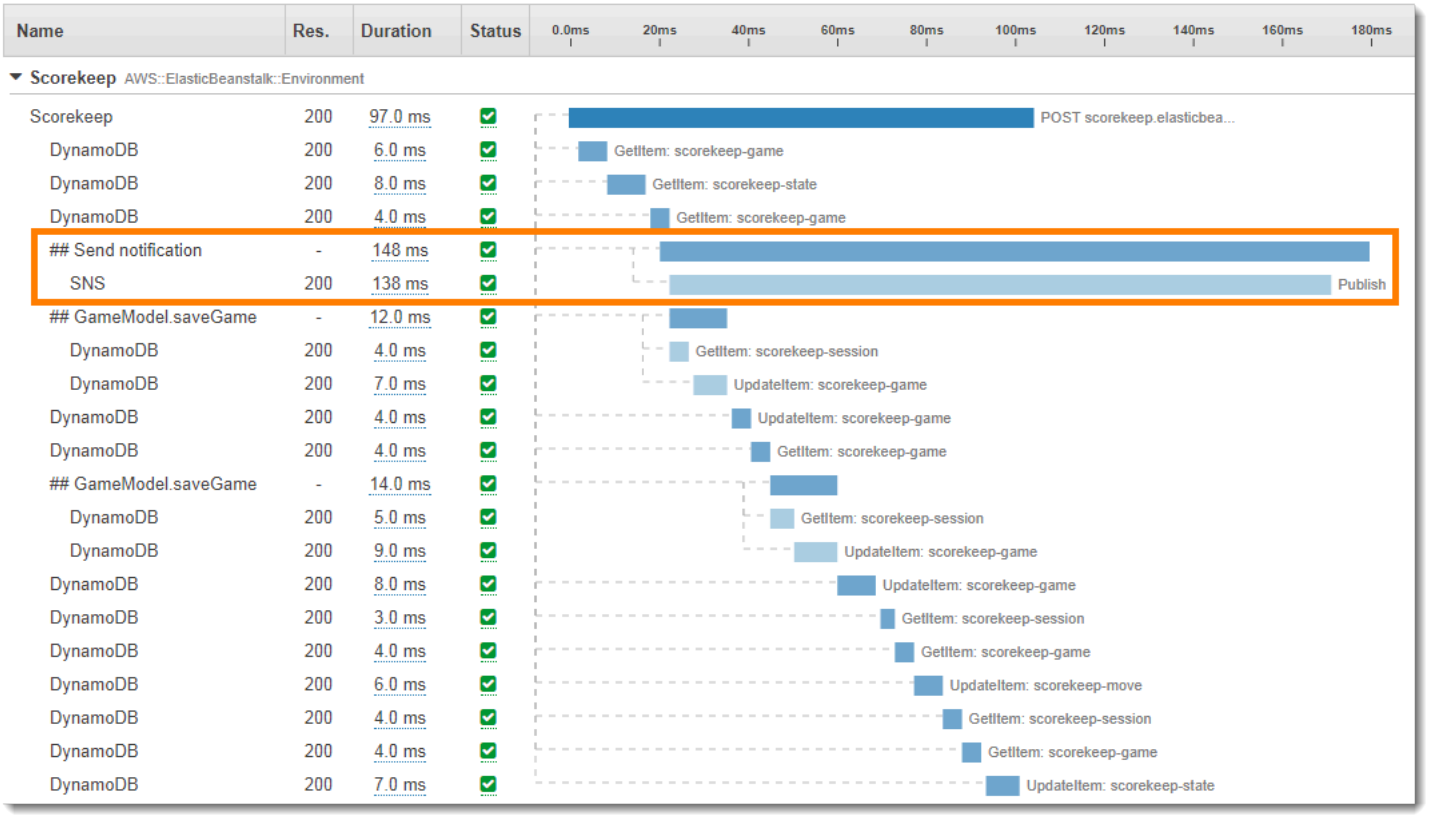
```
Exception in thread "Thread-2" com.amazonaws.xray.exceptions.SegmentNotFoundException:
 Failed to begin subsegment named 'AmazonSNS': segment cannot be found.
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at
 sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at
 sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
...
```

Para solucionar este problema, la aplicación utiliza `GetTraceEntity` para obtener una referencia al segmento en el subproceso principal y `Entity.run()` para ejecutar código del subproceso de trabajo con acceso al contexto del segmento.

Example **src/main/java/scorekeep/MoveFactory.java**: transferencia de contexto del rastro a un subproceso de trabajo

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorder;
import com.amazonaws.xray.entities.Entity;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
      Entity segment = recorder.getTraceEntity();
      Thread comm = new Thread() {
        public void run() {
         segment.run(() -> {
            Subsegment subsegment = AWSXRay.beginSubsegment("## Send notification");
            Sns.sendNotification("Scorekeep game completed", "Winner: " + userId);
            AWSXRay.endSubsegment();
          }
        }
```

Dado que la solicitud ya se ha resuelto antes de la llamada a Amazon SNS, la aplicación crea un subsegmento independiente para el subproceso. Esto impide que el SDK de X-Ray cierre el segmento antes de que registre la respuesta desde Amazon SNS. Si no hay ningún subsegmento abierto cuando Scorekeep resuelve la solicitud, se podría perder la respuesta de Amazon SNS.



Consulte Transmisión de contexto de segmento entre subprocesos en una aplicación multiproceso para obtener más información acerca de los subprocesos múltiples.

AWS X-Ray demonio

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

> **Note**
>
> Ahora puedes usar el CloudWatch agente para recopilar métricas, registros y seguimientos de las EC2 instancias de Amazon y los servidores locales. CloudWatch La versión 1.300025.0 del agente y posteriores puede recopilar rastros de nuestro cliente de OpenTelemetryX-Ray y SDKs enviarlos a X-Ray. Utilizar el CloudWatch agente en lugar del Collector AWS Distro for OpenTelemetry (ADOT) o el daemon X-Ray para recopilar rastros puede ayudarle a reducir la cantidad de agentes que administra. Consulte el tema sobre los CloudWatch agentes en la Guía del CloudWatch usuario para obtener más información.

El AWS X-Ray daemon es una aplicación de software que escucha el tráfico en el puerto UDP 2000, recopila datos de segmentos sin procesar y los transmite a la API. AWS X-Ray El daemon funciona en conjunto con el servicio X-Ray AWS X-Ray SDKs y debe estar ejecutándose para que los datos enviados por el SDKs puedan llegar al servicio de rayos X. El daemon de X-Ray es un proyecto de código abierto. Puedes seguir el proyecto y enviar incidencias y solicitudes de cambios en GitHub: github. com/aws/aws-xray-daemon

Activa AWS Lambda y AWS Elastic Beanstalk usa la integración de esos servicios con X-Ray para ejecutar el daemon. Lambda ejecuta el daemon automáticamente cuando se invoca una función para una solicitud de muestreo. En Elastic Beanstalk , utilice la opción de configuración `XRayEnabled` para ejecutar el daemon en las instancias de su entorno. Para obtener más información, consulte

Para ejecutar el daemon de X-Ray de forma local, local o en otro lugar, descárguelo Servicios de AWS, ejecútelo y, a continuación, dele permiso para cargar documentos segmentados en X-Ray.

Descargar el demonio

Puede descargar el daemon de Amazon S3, Amazon ECR o Docker Hub y, a continuación, ejecutarlo localmente o instalarlo en una EC2 instancia de Amazon en el momento del lanzamiento.

Amazon S3

Instaladores y ejecutables del daemon de X-Ray

- Linux (ejecutable): `aws-xray-daemon-linux-3.x.zip` (sig.)
- Linux (instalador RPM): `aws-xray-daemon-3.x.rpm`
- Linux (instalador DEB): `aws-xray-daemon-3.x.deb`
- Linux (ARM64, ejecutable) — **`aws-xray-daemon-linux-arm64-3.x.zip`**(sig)
- Linux (ARM64, instalador RPM) — **`aws-xray-daemon-arm64-3.x.rpm`**
- Linux (ARM64, instalador DEB) — **`aws-xray-daemon-arm64-3.x.deb`**
- OS X (ejecutable): `aws-xray-daemon-macos-3.x.zip` (sig.)
- Windows (ejecutable): `aws-xray-daemon-windows-process-3.x.zip` (sig.)
- Windows (servicio): `aws-xray-daemon-windows-service-3.x.zip` (sig.)

Estos enlaces siempre apuntan a la última versión 3.x del daemon. Para descargar una versión en concreto, haga lo siguiente:

- Si desea descargar una versión anterior a la versión `3.3.0`, sustituya `3.x` por el número de versión. Por ejemplo, `2.1.0`. Antes de la versión `3.3.0`, la única arquitectura disponible es `arm64`. Por ejemplo, `2.1.0` y `arm64`.
- Si desea descargar una versión posterior a la versión `3.3.0`, sustituya `3.x` por el número de versión y `arch` por el tipo de arquitectura.

Los recursos de X-Ray se replican en buckets de las regiones admitidas. Para usar el depósito más cercano a usted o a sus AWS recursos, sustituya la región de los enlaces anteriores por su región.

```
https://s3.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-daemon/aws-xray-daemon-3.x.rpm
```

Amazon ECR

A partir de la versión 3.2.0, el daemon se encuentra en Amazon ECR. Antes de extraer una imagen, debe autenticar su cliente de Docker en el registro público de Amazon ECR.

Extraiga la etiqueta de la última versión 3.x publicada ejecutando el siguiente comando:

```
docker pull public.ecr.aws/xray/aws-xray-daemon:3.x
```

Las versiones anteriores o alfa se pueden descargar sustituyendo `3.x` por `alpha` o un número de versión específico. No es recomendable utilizar una imagen de daemon con una etiqueta alfa en un entorno de producción.

Docker Hub

El daemon se encuentra en Docker Hub. Para descargar la última versión 3.x publicada ejecutando el siguiente comando:

```
docker pull amazon/aws-xray-daemon:3.x
```

Se pueden lanzar versiones anteriores del daemon sustituyendo `3.x` por la versión deseada.

Verificación de la firma del archivo de demonio

Se incluyen los archivos signature de GPG de los recursos de demonio comprimidos en archivos ZIP. La clave pública es: `aws-xray.gpg`.

Puede utilizar la clave pública para verificar que el archivo ZIP del demonio es original y no se ha modificado. En primer lugar, importe la clave pública con GnuPG.

Para importar la clave pública

1. Descargue la clave pública.

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
```

```
$ wget $BUCKETURL/xray-daemon/aws-xray.gpg
```

2. Importe la clave pública en su llavero.

```
$ gpg --import aws-xray.gpg
gpg: /Users/me/.gnupg/trustdb.gpg: trustdb created
gpg: key 7BFE036BFE6157D3: public key "AWS X-Ray <aws-xray@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Utilice la clave importada para verificar la firma del archivo ZIP del demonio.

Para verificar la firma de un archivo

1. Descargue el archivo y el archivo de firma.

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
$ wget $BUCKETURL/xray-daemon/aws-xray-daemon-linux-3.x.zip
$ wget $BUCKETURL/xray-daemon/aws-xray-daemon-linux-3.x.zip.sig
```

2. Ejecute `gpg --verify` para verificar la firma.

```
$ gpg --verify aws-xray-daemon-linux-3.x.zip.sig aws-xray-daemon-linux-3.x.zip
gpg: Signature made Wed 19 Apr 2017 05:06:31 AM UTC using RSA key ID FE6157D3
gpg: Good signature from "AWS X-Ray <aws-xray@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EA6D 9271 FBF3 6990 277F  4B87 7BFE 036B FE61 57D3
```

Tenga en cuenta la advertencia sobre confianza. Una clave solo es de confianza si la ha firmado usted o alguien en quien confíe. Esto no significa que la firma no sea válida, solo que no han verificado la clave pública.

Ejecutar el demonio

Ejecute el demonio localmente desde la línea de comandos. Utilice la opción `-o` para ejecutarlo en modo local y `-n` para configurar la región.

```
~/Downloads$ ./xray -o -n us-east-2
```

Si desea obtener instrucciones específicas para cada plataforma, consulte estos temas:

- Linux (local): Ejecución del daemon de X-Ray en Linux
- Windows (local): Ejecución de un daemon de X-Ray en Windows
- Elastic Beanstalk: Ejecutar el daemon X-Ray en AWS Elastic Beanstalk
- Amazon EC2 — Ejecutar el daemon X-Ray en Amazon EC2
- Amazon ECS: Ejecución del daemon de X-Ray en Amazon ECS

Puede personalizar aún más el comportamiento del demonio utilizando las opciones de la línea de comandos o un archivo de configuración. Para obtener más información, consulte Configuración del AWS X-Ray daemon.

Permiso para el envío de datos a X-Ray desde el daemon

El daemon de X-Ray usa el AWS SDK para cargar datos de rastreo en X-Ray y necesita AWS credenciales con permiso para hacerlo.

En Amazon EC2, el daemon usa el rol de perfil de instancia de la instancia automáticamente. Para obtener información sobre las credenciales necesarias para ejecutar el daemon de forma local, consulte Ejecutar la aplicación de forma local.

Si especifica las credenciales en más de una ubicación (archivo de credenciales, perfil de instancia o variables de entorno), la cadena de proveedores del SDK determina qué credenciales se utilizan. Para obtener más información acerca de cómo proporcionar credenciales al SDK, consulte la sección sobre especificación de credenciales en la Guía para desarrolladores del SDK de AWS para Go.

El rol o usuario de IAM al cual pertenecen las credenciales del demonio deben tener el permiso de escribir datos en el servicio de forma automática.

- Para usar el daemon en Amazon EC2, cree un nuevo rol de perfil de instancia o añada la política administrada a una existente.
- Para utilizar el daemon en Elastic Beanstalk, añada la política administrada al rol predeterminado del perfil de instancia de Elastic Beanstalk.
- Para ejecutar el daemon de forma local, consulte Ejecutar la aplicación de forma local.

Para obtener más información, consulte Administración de identidad y acceso para AWS X-Ray.

Registros del daemon de X-Ray

El daemon genera información sobre su configuración actual y los segmentos a los que envía. AWS X-Ray

```
2016-11-24T06:07:06Z [Info] Initializing AWS X-Ray daemon 2.1.0
2016-11-24T06:07:06Z [Info] Using memory limit of 49 MB
2016-11-24T06:07:06Z [Info] 313 segment buffers allocated
2016-11-24T06:07:08Z [Info] Successfully sent batch of 1 segments (0.123 seconds)
2016-11-24T06:07:09Z [Info] Successfully sent batch of 1 segments (0.006 seconds)
```

De forma predeterminada, el daemon envía los logs a STDOUT. Si ejecuta el demonio en segundo plano, utilice la opción de línea de comandos `--log-file` o un archivo de configuración para establecer la ruta del archivo de log. También puede definir el nivel de log y deshabilitar la rotación de logs. Para obtener instrucciones, consulte Configuración del AWS X-Ray daemon.

En Elastic Beanstalk, la plataforma establece la ubicación de los registros del daemon. Para obtener más información, consulte Ejecutar el daemon X-Ray en AWS Elastic Beanstalk.

Configuración del AWS X-Ray daemon

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede utilizar las opciones de la línea de comandos o un archivo de configuración para personalizar el comportamiento del daemon de X-Ray. La mayoría de las opciones están disponibles con ambos métodos, pero algunas solo están disponibles en los archivos de configuración y otras solo en la línea de comandos.

Para empezar, la única opción que debe conocer es `-n` o `--region`, que se utiliza para establecer la región que el daemon utiliza para enviar los datos de rastro a X-Ray.

```
~/xray-daemon$ ./xray -n us-east-2
```

Si ejecuta el daemon localmente, es decir, no en Amazon EC2, puede añadir la `-o` opción de omitir la comprobación de las credenciales del perfil de la instancia para que el daemon esté listo más rápidamente.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

El resto de las opciones de la línea de comandos le permiten configurar el registro, escuchar en otro puerto, limitar la cantidad de memoria que puede utilizar el demonio o asumir un rol para enviar datos de rastreo a otra cuenta.

Puede pasar un archivo de configuración al daemon para tener acceso a las opciones avanzadas de configuración y realizar tareas como limitar el número de llamadas simultáneas a X-Ray, deshabilitar la rotación de registros, y enviar tráfico a un proxy.

Secciones

Variables de entorno admitidas

El daemon de X-Ray admite las siguientes variables de entorno:

- `AWS_REGION`: especifica la Región de AWS del punto de conexión del servicio de X-Ray.
- `HTTPS_PROXY`: especifica una dirección proxy a través de la cual el daemon puede cargar los segmentos. Se pueden utilizar un nombre de dominio DNS o una dirección IP y los números de puerto que utilizan los servidores proxy.

Uso de las opciones de la línea de comandos

Pase estas opciones al demonio cuando lo ejecute localmente o con un script de datos de usuario.

Opciones de la línea de comandos

- `-b`, `--bind`: buscan documentos de segmento en un puerto UDP diferente.

```
--bind "127.0.0.1:3000"
```

Valor predeterminado: `2000`.

- `-t`, `--bind-tcp`: escuchan las llamadas al servicio de X-Ray en un puerto TCP diferente.

```
-bind-tcp "127.0.0.1:3000"
```

Valor predeterminado: `2000`.

- `-c`, `--config`: cargan un archivo de configuración desde la ruta especificada.

```
--config "/home/ec2-user/xray-daemon.yaml"
```

- `-f`, `--log-file`: envían registros a ruta de archivo especificada.

```
--log-file "/var/log/xray-daemon.log"
```

- `-l`, `--log-level`: muestran el nivel de registro, desde el más detallado al menos detallado: dev, debug, info, warn, error y prod.

```
--log-level warn
```

Valor predeterminado: `prod`.

- `-m`, `--buffer-memory`: cambian la cantidad de memoria en MB que pueden utilizar los búferes (mínimo 3).

```
--buffer-memory 50
```

Predeterminado: 1 % de memoria disponible

- `-o`, `--local-mode` — No compruebes, por EC2 ejemplo, los metadatos.
- `-r`, `--role-arn`: asumen el rol de IAM especificado para cargar segmentos en una cuenta diferente.

```
--role-arn "arn:aws:iam::123456789012:role/xray-cross-account"
```

- `-a`, `--resource-arn` — Nombre de recurso de Amazon (ARN) del AWS recurso que ejecuta el daemon.

- `-p`, `--proxy-address` — Cargue segmentos a AWS X-Ray través de un proxy. Hay que especificar el protocolo del servidor proxy.

```
--proxy-address "http://192.0.2.0:3000"
```

- `-n`, `--region`: envían segmentos al servicio de X-Ray en una región específica.
- `-v`, `--version` — Muestra la versión AWS X-Ray del daemon.
- `-h`, `--help`: muestran la pantalla de ayuda.

Uso de un archivo de configuración

También puede utilizar un archivo con formato YAML para configurar el demonio. Transfiera el archivo de configuración al demonio mediante la opción `-c`.

```
~$ ./xray -c ~/xray-daemon.yaml
```

Configuración de las opciones de archivo

- `TotalBufferSizeMB`: tamaño máximo del búfer en MB (mínimo 3). Elija 0 para utilizar 1% de memoria del host.
- `Concurrency`— Número máximo de llamadas simultáneas AWS X-Ray para cargar documentos segmentados.
- `Region`— Enviar segmentos al AWS X-Ray servicio de una región específica.
- `Socket`: configura el enlace del daemon.
  - `UDPAddress`: cambia el puerto en el que el daemon escucha.
  - `TCPAddress`: escucha las llamadas al servicio de X-Ray en un puerto TCP diferente.
- `Logging`: configura el comportamiento de registro.
  - `LogRotation`: se establece en `false` para deshabilitar la rotación de registros.
  - `LogLevel`: cambia el nivel de registro, desde el más detallado al menos: `dev`, `debug`, `info` o `prod`, `warn`, `error`, y `prod`. El valor predeterminado es `prod`, que equivale a `info`.
  - `LogPath`: envía los registros a la ruta de archivo especificada.
- `LocalMode`— Configúrelo `true` para omitir la comprobación de los metadatos de la EC2 instancia.
- `ResourceARN`— Nombre de recurso de Amazon (ARN) del AWS recurso que ejecuta el daemon.

- `RoleARN`: asume el rol de IAM especificado para cargar segmentos en una cuenta diferente.
- `ProxyAddress`— Cargue segmentos a AWS X-Ray través de un proxy.
- `Endpoint`: cambia el punto de conexión del servicio de X-Ray al que el daemon envía documentos de segmento.
- `NoVerifySSL`: desactiva la verificación del certificado TLS.
- `Version`: versión de formato del archivo de configuración del daemon. La versión del formato del archivo es un campo obligatorio.

Example Xray-daemon.yaml

Este archivo de configuración cambia el puerto de escucha del demonio a 3000, desactiva las comprobaciones de metadatos de instancias, establece el rol que se usará para cargar segmentos y cambia las opciones de región y registro.

```
Socket:
  UDPAddress: "127.0.0.1:3000"
  TCPAddress: "127.0.0.1:3000"
Region: "us-west-2"
Logging:
  LogLevel: "warn"
  LogPath: "/var/log/xray-daemon.log"
LocalMode: true
RoleARN: "arn:aws:iam::123456789012:role/xray-cross-account"
Version: 2
```

Ejecución del daemon de X-Ray localmente

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede ejecutar el AWS X-Ray daemon localmente en Linux, macOS, Windows o en un contenedor de Docker. Ejecute el daemon para transmitir los datos de rastro a X-Ray durante el desarrollo y las pruebas de una aplicación instrumentada. Descargue y extraiga el daemon mediante las instrucciones que encontrará aquí.

Cuando se ejecuta de forma local, el daemon puede leer las credenciales de un archivo de credenciales del AWS SDK (`.aws/credentials`en su directorio de usuarios) o de variables de entorno. Para obtener más información, consulte Permiso para el envío de datos a X-Ray desde el daemon.

El demonio escucha los datos UDP en el puerto 2000. Puede cambiar el puerto y otras opciones mediante un archivo de configuración y distintas opciones de línea de comandos. Para obtener más información, consulte Configuración del AWS X-Ray daemon.

Ejecución del daemon de X-Ray en Linux

Puede ejecutar el daemon ejecutable desde la línea de comandos. Utilice la opción `-o` para ejecutarlo en modo local y `-n` para configurar la región.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Para ejecutar el demonio en segundo plano, use &.

```
~/xray-daemon$ ./xray -o -n us-east-2 &
```

Finalice un proceso de demonio que se ejecuta en segundo plano con `pkill`.

```
~$ pkill xray
```

Ejecución del daemon de X-Ray en un contenedor de Docker

Para ejecutar el demonio localmente en un contenedor de Docker, guarde el texto siguiente en un archivo denominado `Dockerfile`. Descargue la imagen de ejemplo completa en Amazon ECR. Para obtener más información, consulte Descarga del daemon.

Example Dockerfile: Amazon Linux

```
FROM amazonlinux
```

```
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

Compile la imagen del contenedor con `docker build`.

```
~/xray-daemon$ docker build -t xray-daemon .
```

Ejecute la imagen en un contenedor con `docker run`.

```
~/xray-daemon$ docker run \
      --attach STDOUT \
      -v ~/.aws/:/root/.aws/:ro \
      --net=host \
      -e AWS_REGION=us-east-2 \
      --name xray-daemon \
      -p 2000:2000/udp \
      xray-daemon -o
```

Este comando utiliza las siguientes opciones:

- `--attach STDOUT`: ver la salida del daemon en el terminal.
- `-v ~/.aws/:/root/.aws/:ro`— Conceda al contenedor acceso de solo lectura al `.aws` directorio para que pueda leer sus AWS credenciales del SDK.
- `AWS_REGION=`*`us-east-2`*: establecer la variable de entorno `AWS_REGION` para indicar al daemon la región que debe utilizar.
- `--net=host`: adjunte el contenedor a la red `host`. Los contenedores de la red host pueden comunicarse entre sí sin publicar puertos.
- `-p 2000:2000/udp`: asignar el puerto UDP 2000 en el equipo al mismo puerto del contenedor. Esto no es necesario para que se comuniquen los contenedores de la misma red, pero sí permite enviar segmentos al demonio desde la línea de comandos o desde una aplicación que no se ejecuta en Docker.
- `--name xray-daemon`: asignar el nombre `xray-daemon` al contenedor en lugar de generar un nombre aleatorio.

- `-o` (después del nombre de la imagen): añadir la opción `-o` al punto de entrada que ejecuta el daemon en el contenedor. Esta opción indica al daemon que se ejecute en modo local para evitar que intente leer los metadatos de las EC2 instancias de Amazon.

Para detener el demonio, utilice `docker stop`. Si realiza cambios en el archivo `Dockerfile` y crea una imagen nueva, debe eliminar el contenedor existente para poder crear otra con el mismo nombre. Utilice `docker rm` para eliminar el contenedor.

```
$ docker stop xray-daemon
$ docker rm xray-daemon
```

Ejecución de un daemon de X-Ray en Windows

Puede ejecutar el daemon ejecutable desde la línea de comandos. Utilice la opción `-o` para ejecutarlo en modo local y `-n` para configurar la región.

```
> .\xray_windows.exe -o -n us-east-2
```

Utilice un PowerShell script para crear y ejecutar un servicio para el daemon.

Example PowerShell secuencia de comandos: Windows

```
if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ){
    sc.exe stop AWSXRayDaemon
    sc.exe delete AWSXRayDaemon
}
if ( Get-Item -path aws-xray-daemon -ErrorAction SilentlyContinue ) {
    Remove-Item -Recurse -Force aws-xray-daemon
}

$currentLocation = Get-Location
$zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
$zipPath = "$currentLocation\$zipFileName"
$destPath = "$currentLocation\aws-xray-daemon"
$daemonPath = "$destPath\xray.exe"
$daemonLogPath = "C:\inetpub\wwwroot\xray-daemon.log"
$url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-
daemon/aws-xray-daemon-windows-service-3.x.zip"

Invoke-WebRequest -Uri $url -OutFile $zipPath
```

```
Add-Type -Assembly "System.IO.Compression.Filesystem"
[io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

sc.exe create AWSXRayDaemon binPath= "$daemonPath -f $daemonLogPath"
sc.exe start AWSXRayDaemon
```

Ejecución del daemon de X-Ray en OS X

Puede ejecutar el daemon ejecutable desde la línea de comandos. Utilice la opción `-o` para ejecutarlo en modo local y `-n` para configurar la región.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2
```

Para ejecutar el demonio en segundo plano, use &.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2 &
```

Utilice nohup para evitar que el demonio finalice cuando se cierre la terminal.

```
~/xray-daemon$ nohup ./xray_mac &
```

Ejecutar el daemon X-Ray en AWS Elastic Beanstalk

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Para retransmitir los datos de rastreo de la aplicación a ella AWS X-Ray, puede ejecutar el daemon X-Ray en las instancias de Amazon del entorno de Elastic Beanstalk. EC2 Para ver una lista de plataformas compatibles, consulte Configuración de la depuración en AWS X-Ray en la Guía para desarrolladores de AWS Elastic Beanstalk .

> **Note**
>
> El daemon utiliza el perfil de instancia del entorno para obtener permisos. Para obtener instrucciones sobre cómo añadir permisos al perfil de instancia de Elastic Beanstalk, consulte Permiso para el envío de datos a X-Ray desde el daemon.

Las plataformas de Elastic Beanstalk proporcionan una opción de configuración que puede establecer para ejecutar el daemon de forma automática. Puede habilitar el daemon en un archivo de configuración en el código fuente o mediante la opción disponible en la consola de Elastic Beanstalk. Cuando habilita la opción de configuración, el demonio se instala en la instancia y se ejecuta como servicio.

Es posible que la versión incluida en las plataformas de Elastic Beanstalk no sea la versión más reciente. Consulte Plataformas compatibles para ver qué versión del demonio está disponible para la configuración de la plataforma que utiliza.

Elastic Beanstalk no proporciona el daemon X-Ray en la plataforma Multicontainer Docker (Amazon ECS).

Uso de la integración de X-Ray de Elastic Beanstalk para ejecutar el daemon de X-Ray

Utilice la consola para activar la integración de X-Ray o configúrela en el código fuente de la aplicación con un archivo de configuración.

Para habilitar el daemon de X-Ray en la consola de Elastic Beanstalk:

1. Abra la consola de Elastic Beanstalk.
2. Desplácese hasta la consola de administración del entorno.
3. Elija Configuración.
4. Elija Software Settings (Configuración de software).
5. En X-Ray daemon (Demonio de X-Ray), elija Enabled (Habilitado).
6. Seleccione Aplicar.

Puede incluir un archivo de configuración en el código fuente para que la configuración sea portátil entre entornos.

Example.ebextensions/xray-daemon.config

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

Elastic Beanstalk transfiere un archivo de configuración al daemon y envía los registros a una ubicación estándar.

En plataformas Windows Server

- Archivo de configuración: `C:\Program Files\Amazon\XRay\cfg.yaml`
- Registros: `c:\Program Files\Amazon\XRay\logs\xray-service.log`

En plataformas Linux

- Archivo de configuración: `/etc/amazon/xray/cfg.yaml`
- Registros: `/var/log/xray/xray.log`

Elastic Beanstalk proporciona herramientas para extraer registros Consola de administración de AWS de instancias desde la línea de comandos o. Puede indicar a Elastic Beanstalk que incluya los registros del daemon de X-Ray añadiendo una tarea con un archivo de configuración.

Example.ebextensions/xray-logs.config - Linux

```
files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      /var/log/xray/xray.log
```

Example.ebextensions/xray-logs.config: Windows Server

```
files:
  "c:/Program Files/Amazon/ElasticBeanstalk/config/taillogs.d/xray-daemon.conf" :
    mode: "000644"
```

```
    owner: root
    group: root
    content: |
      c:\Progam Files\Amazon\XRay\logs\xray-service.log
```

Consulte Visualización de registros de las AWS Elastic Beanstalk instancias de Amazon del entorno de Elastic Beanstalk en EC2 la Guía para desarrolladores para obtener más información.

Descarga y ejecución del daemon de X-Ray manualmente (avanzado)

Si el daemon de X-Ray no está disponible para la configuración de la plataforma, puede descargarlo desde Amazon S3 y ejecutarlo con un archivo de configuración.

Utilice un archivo de configuración de Elastic Beanstalk para descargar y ejecutar el daemon.

Example.ebextensions/xray.config - Linux

```
commands:
  01-stop-tracing:
    command: yum remove -y xray
    ignoreErrors: true
  02-copy-tracing:
    command: curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-
daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
  03-start-tracing:
    command: yum install -y /home/ec2-user/xray.rpm

files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      /var/log/xray/xray.log
  "/etc/amazon/xray/cfg.yaml" :
    mode: "000644"
    owner: root
    group: root
    content: |
      Logging:
        LogLevel: "debug"
      Version: 2
```

Example.ebextensions/xray.config: Windows Server

```
container_commands:
  01-execute-config-script:
    command: Powershell.exe -ExecutionPolicy Bypass -File c:\\temp\\installDaemon.ps1
    waitAfterCompletion: 0

files:
  "c:/temp/installDaemon.ps1":
    content: |
      if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ) {
          sc.exe stop AWSXRayDaemon
          sc.exe delete AWSXRayDaemon
      }

      $targetLocation = "C:\Program Files\Amazon\XRay"
      if ((Test-Path $targetLocation) -eq 0) {
          mkdir $targetLocation
      }

      $zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
      $zipPath = "$targetLocation\$zipFileName"
      $destPath = "$targetLocation\aws-xray-daemon"
      if ((Test-Path $destPath) -eq 1) {
          Remove-Item -Recurse -Force $destPath
      }

      $daemonPath = "$destPath\xray.exe"
      $daemonLogPath = "$targetLocation\xray-daemon.log"
      $url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/
xray-daemon/aws-xray-daemon-windows-service-3.x.zip"

      Invoke-WebRequest -Uri $url -OutFile $zipPath
      Add-Type -Assembly "System.IO.Compression.Filesystem"
      [io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

      New-Service -Name "AWSXRayDaemon" -StartupType Automatic -BinaryPathName
 "`"$daemonPath`" -f `"$daemonLogPath`""
      sc.exe start AWSXRayDaemon
    encoding: plain
  "c:/Program Files/Amazon/ElasticBeanstalk/config/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
```

```
    content: |
      C:\Program Files\Amazon\XRay\xray-daemon.log
```

Estos ejemplos también añaden el archivo de registro del daemon a la tarea de registros de finalización de Elastic Beanstalk para incluirlo cuando se soliciten registros con la consola o la interfaz de línea de comandos de Elastic Beanstalk (CLI de EB).

Ejecutar el daemon X-Ray en Amazon EC2

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede ejecutar el daemon X-Ray en los siguientes sistemas operativos de Amazon EC2:

- Amazon Linux
- Ubuntu
- Windows Server (2012 R2 y posteriores)

Utilice un perfil de instancia para conceder permiso al daemon para cargar datos de rastro en X-Ray. Para obtener más información, consulte Permiso para el envío de datos a X-Ray desde el daemon.

Utilice un script de datos de usuario para ejecutar el demonio de manera automática cuando inicie la instancia.

Example Script de datos de usuario: Linux

```
#!/bin/bash
curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
yum install -y /home/ec2-user/xray.rpm
```

Example Script de datos de usuario: Windows Server

```
<powershell>
if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ) {
    sc.exe stop AWSXRayDaemon
    sc.exe delete AWSXRayDaemon
}

$targetLocation = "C:\Program Files\Amazon\XRay"
if ((Test-Path $targetLocation) -eq 0) {
    mkdir $targetLocation
}

$zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
$zipPath = "$targetLocation\$zipFileName"
$destPath = "$targetLocation\aws-xray-daemon"
if ((Test-Path $destPath) -eq 1) {
    Remove-Item -Recurse -Force $destPath
}

$daemonPath = "$destPath\xray.exe"
$daemonLogPath = "$targetLocation\xray-daemon.log"
$url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-
daemon/aws-xray-daemon-windows-service-3.x.zip"

Invoke-WebRequest -Uri $url -OutFile $zipPath
Add-Type -Assembly "System.IO.Compression.Filesystem"
[io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

New-Service -Name "AWSXRayDaemon" -StartupType Automatic -BinaryPathName
 "`"$daemonPath`" -f `"$daemonLogPath`""
sc.exe start AWSXRayDaemon
</powershell>
```

Ejecución del daemon de X-Ray en Amazon ECS

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener

más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

En Amazon ECS, cree una imagen de Docker que ejecute el daemon de X-Ray, cárguelo en el repositorio de imágenes de Docker y luego impleméntelo en su clúster de Amazon ECS. Puede utilizar mapeos de puertos y la configuración del modo de red en el archivo de definición de tareas para permitir que la aplicación se comunique con el contenedor del demonio.

Uso de la imagen de Docker oficial de la

X-Ray proporciona una imagen del contenedor de Docker en Amazon ECR que puede implementar junto con la aplicación. Para obtener más información, consulte Descarga del daemon.

Example Definición de tarea

```
{
  "name": "xray-daemon",
  "image": "amazon/aws-xray-daemon",
  "cpu": 32,
  "memoryReservation": 256,
  "portMappings" : [
      {
          "hostPort": 0,
          "containerPort": 2000,
          "protocol": "udp"
      }
   ]
}
```

Creación y compilación de una imagen de Docker

Si desea realizar una configuración personalizada, es posible que tenga que definir su propia imagen de Docker.

Añada políticas administradas al rol de tarea para conceder permiso al daemon para cargar datos de rastro en X-Ray. Para obtener más información, consulte Permiso para el envío de datos a X-Ray desde el daemon.

Utilice uno de los siguientes Dockerfiles para crear una imagen que ejecute el demonio.

Example Dockerfile: Amazon Linux

```
FROM amazonlinux
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

> **Note**
>
> Las marcas `-t` y `-b` son necesarias para especificar una dirección de enlace que escuche el bucle invertido de un entorno con varios contenedores.

Example Dockerfile: Ubuntu

Para los derivados de Debian, tendrá que instalar certificados de una autoridad de certificación (CA) para evitar problemas al descargar el instalador.

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y --force-yes --no-install-recommends apt-
transport-https curl ca-certificates wget && apt-get clean && apt-get autoremove && rm
 -rf /var/lib/apt/lists/*
RUN wget https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-
xray-daemon-3.x.deb
RUN dpkg -i aws-xray-daemon-3.x.deb
ENTRYPOINT ["/usr/bin/xray", "--bind=0.0.0.0:2000", "--bind-tcp=0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

En la definición de tarea, la configuración depende del modo de red que se utilice. El modo de red puente es la opción predeterminada y se puede utilizar en la VPC predeterminada. En una red en modo puente, establezca la variable de entorno `AWS_XRAY_DAEMON_ADDRESS` para indicar al SDK de X-Ray a qué puerto del contenedor se ha de hacer referencia y establezca el puerto de host. Por ejemplo, podría publicar el puerto UDP 2000 y crear un enlace desde el contenedor de la aplicación hasta el contenedor del demonio.

Example Definición de tarea

```
    {
      "name": "xray-daemon",
      "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
      "cpu": 32,
      "memoryReservation": 256,
      "portMappings" : [
          {
              "hostPort": 0,
              "containerPort": 2000,
              "protocol": "udp"
          }
       ]
    },
    {
      "name": "scorekeep-api",
      "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
      "cpu": 192,
      "memoryReservation": 512,
      "environment": [
          { "name" : "AWS_REGION", "value" : "us-east-2" },
          { "name" : "NOTIFICATION_TOPIC", "value" : "arn:aws:sns:us-
east-2:123456789012:scorekeep-notifications" },
          { "name" : "AWS_XRAY_DAEMON_ADDRESS", "value" : "xray-daemon:2000" }
      ],
      "portMappings" : [
          {
              "hostPort": 5000,
              "containerPort": 5000
          }
      ],
      "links": [
        "xray-daemon"
      ]
    }
```

Si ejecuta el clúster en la subred privada de una VPC, puede utilizar el modo de red `awsvpc` para asociar una interfaz de red elástica (ENI) a los contenedores. De este modo, puede evitar utilizar enlaces. Omita el puerto de host en los mapeos de puertos, el enlace y la variable de entorno `AWS_XRAY_DAEMON_ADDRESS`.

Example Definición de tarea de VPC

```
{
    "family": "scorekeep",
    "networkMode":"awsvpc",
    "containerDefinitions": [
        {
          "name": "xray-daemon",
          "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
          "cpu": 32,
          "memoryReservation": 256,
          "portMappings" : [
              {
                  "containerPort": 2000,
                  "protocol": "udp"
              }
          ]
        },
        {
            "name": "scorekeep-api",
            "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
            "cpu": 192,
            "memoryReservation": 512,
            "environment": [
                { "name" : "AWS_REGION", "value" : "us-east-2" },
                { "name" : "NOTIFICATION_TOPIC", "value" : "arn:aws:sns:us-
east-2:123456789012:scorekeep-notifications" }
            ],
            "portMappings" : [
                {
                    "containerPort": 5000
                }
            ]
        }
    ]
}
```

Configuración de las opciones de línea de comandos en la consola de Amazon ECS

Las opciones de línea de comandos anulan cualquier valor que presente algún conflicto del archivo de configuración de la imagen. Las opciones de línea de comandos se suelen utilizar para pruebas

locales, pero también se pueden utilizar para mayor comodidad al establecer variables de entorno o con el fin de controlar el proceso de inicio.

Al añadir opciones de línea de comandos, se actualiza el `CMD` de Docker que se pasa al contenedor. Para obtener más información, consulte Docker run reference.

Para establecer una opción de línea de comandos

1. Abra la consola clásica de Amazon ECS en https://console.aws.amazon.com/ecs/.
2. En la barra de navegación, seleccione la región que contiene la definición de tarea.
3. En el panel de navegación, elija Task Definitions.
4. En la página Task Definitions, seleccione la casilla situada a la izquierda de la definición de tarea que revisar y seleccione Create new revision.
5. En la página Create new revision of Task Definition (Crear nueva revisión de definición de tarea), seleccione el contenedor.
6. En la sección ENVIRONMENT (ENTORNO), añada al campo Command (Comando) la lista de opciones de línea de comandos separadas por comas.
7. Elija Actualizar.
8. Verifique la información y seleccione Create.

En el ejemplo siguiente se muestra cómo escribir una opción de línea de comandos separada por comas para la opción `RoleARN`. La opción `RoleARN` asume el rol de IAM especificado para cargar segmentos en una cuenta diferente.

Example

```
--role-arn, arn:aws:iam::123456789012:role/xray-cross-account
```

Para obtener más información sobre las opciones de línea de comandos disponibles en X-Ray, consulte Configuración del AWS X-Ray Daemon.

Integración AWS X-Ray con otros Servicios de AWS

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Muchos Servicios de AWS ofrecen distintos niveles de integración de X-Ray, como el muestreo y la adición de encabezados a las solicitudes entrantes, la ejecución del daemon X-Ray y el envío automático de datos de rastreo a X-Ray. La integración en X-Ray puede incluir lo siguiente:

- Instrumentación activa: realiza un muestreo de las solicitudes entrantes y las instrumenta.
- Instrumentación pasiva: instrumenta solicitudes cuyo muestreo lo realizó otro servicio.
- Rastreo de solicitudes: agrega un encabezado de rastreo a todas las solicitudes entrantes y lo propaga.
- Herramientas: ejecuta el daemon de X-Ray para recibir segmentos del SDK de X-Ray.

> **Note**
>
> Los X-Ray SDKs incluyen complementos para una integración adicional con Servicios de AWS. Por ejemplo, puede utilizar el complemento Elastic Beanstalk del SDK de X-Ray para Java con el fin de agregar información acerca del entorno de Elastic Beanstalk que ejecuta su aplicación, incluidos el nombre e ID del entorno.

Estos son algunos ejemplos de los Servicios de AWS que están integrados con X-Ray:

- AWS Distro for OpenTelemetry (ADOT): con ADOT, los ingenieros pueden instrumentar sus aplicaciones una vez y enviar métricas y rastreos correlacionados a varias AWS soluciones de

monitoreo, como Amazon CloudWatch, Amazon Service y AWS X-Ray Amazon OpenSearch Managed Service para Prometheus.

- AWS Lambda— Instrumentación activa y pasiva de las solicitudes entrantes en todos los tiempos de ejecución. AWS Lambda añade dos nodos a su mapa de rastreo, uno para el AWS Lambda servicio y otro para la función. Al habilitar la instrumentación, AWS Lambda también ejecuta el daemon X-Ray en los tiempos de ejecución de Java y Node.js para usarlo con el SDK de X-Ray.
- Amazon API Gateway: instrumentación activa y pasiva. API Gateway aplica reglas de muestreo para determinar qué solicitudes registrar y agrega un nodo para la etapa de puerta de enlace al mapa de servicio.
- AWS Elastic Beanstalk: herramientas. Elastic Beanstalk incluye el daemon de X-Ray en las siguientes plataformas:
  - Java SE: 2.3.0 y configuraciones más recientes
  - Tomcat: 2.4.0 y configuraciones más recientes
  - Node.js: 3.2.0 y configuraciones más recientes
  - Windows Server: todas las configuraciones que no sean Windows Server Core liberadas a partir del 9 de diciembre de 2016.

  Puede usar la consola de Elastic Beanstalk para indicar a Elastic Beanstalk que ejecute el daemon en estas plataformas o utilizar la opción `XRayEnabled` en el espacio de nombres `aws:elasticbeanstalk:xray`.
- Elastic Load Balancing: rastreo de solicitudes en equilibradores de carga de aplicaciones El equilibrador de carga de aplicación agrega el ID de rastro al encabezado de la solicitud antes de enviarlo al grupo de destino.
- Amazon EventBridge — Instrumentación pasiva. Si un servicio que publica eventos EventBridge está equipado con el SDK de X-Ray, los destinos de los eventos recibirán el encabezado de rastreo y podrán seguir propagando el ID de rastreo original.
- Amazon Simple Notification Service: instrumentación pasiva. Si un publicador de Amazon SNS hace un rastreo de su cliente con el SDK de X-Ray, el suscriptor puede recuperar el encabezado de rastreo y seguir propagando el rastro original a partir del publicador con el mismo ID de rastro.
- Amazon Simple Queue Service: instrumentación pasiva. Si un servicio rastrea solicitudes utilizando el SDK de X-Ray, Amazon SQS podrá enviar el encabezado de rastreo y continuar propagando el rastro original entre el remitente y el consumidor con un ID de rastro coherente.
- Amazon Bedrock AgentCore: AgentCore admite el rastreo distribuido mediante la integración de X-Ray, lo que le permite realizar un seguimiento de las solicitudes a medida que fluyen por las

solicitudes de sus agentes. Cuando habilita la observabilidad de sus AgentCore recursos, puede propagar el contexto del rastreo más allá de los límites de los servicios y obtener visibilidad del rendimiento de sus agentes y herramientas de IA.

Elija uno de los siguientes temas para explorar el conjunto completo de temas integrados. Servicios de AWS

Temas

- Amazon Bedrock AgentCore y AWS X-Ray
- Amazon Elastic Compute Cloud y AWS X-Ray
- Amazon SNS y AWS X-Ray
- Amazon SQS y AWS X-Ray
- Amazon S3 y AWS X-Ray
- AWS Distro para OpenTelemetry y AWS X-Ray
- Rastreo de los cambios en la configuración de cifrado de X-Ray con AWS Config
- AWS AppSync y AWS X-Ray
- Compatibilidad de Amazon API Gateway con el rastreo activo para AWS X-Ray
- Amazon EC2 y AWS App Mesh
- AWS App Runner y X-Ray
- Registro de llamadas a la API de X-Ray con AWS CloudTrail
- Integración con CloudWatch con X-Ray
- AWS Elastic Beanstalk y AWS X-Ray
- Elastic Load Balancing y AWS X-Ray
- Amazon EventBridge y AWS X-Ray
- AWS Lambda y AWS X-Ray
- AWS Step Functions y AWS X-Ray

Amazon Bedrock AgentCore y AWS X-Ray

Amazon Bedrock AgentCore se integra AWS X-Ray para proporcionar capacidades de rastreo distribuido a sus agentes y herramientas de IA. Esta integración le permite realizar un seguimiento de las solicitudes a medida que pasan por las aplicaciones de los agentes, lo que le ayuda a identificar los cuellos de botella de rendimiento y a solucionar problemas.

AgentCore admite el rastreo distribuido mediante la integración de X-Ray, lo que le permite supervisar el rendimiento de sus agentes y herramientas de IA. Cuando habilita la observabilidad de sus AgentCore recursos, puede propagar el contexto del rastreo más allá de los límites del servicio y obtener visibilidad de la forma en que sus agentes interactúan con otros servicios. AWS Para obtener más información, consulte Amazon Bedrock AgentCore.

AgentCore admite las siguientes funciones de X-Ray:

- Propagación del contexto del rastro a los servicios posteriores
- Instrumentación personalizada mediante el SDK AWS Distro for OpenTelemetry (ADOT)

Configuración de X-Ray con AgentCore

Para usar X-Ray con AgentCore, debes habilitar la búsqueda de CloudWatch transacciones en tu AWS cuenta. Se trata de una configuración única que permite AgentCore enviar datos de rastreo a X-Ray. Para obtener más información, consulte Habilitación de Transaction Search.

Para obtener más información sobre cómo configurar la observabilidad para AgentCore, consulte Añadir observabilidad a su AgentCore agente o herramienta de Amazon Bedrock.

Uso de encabezados de rastreo con AgentCore

AgentCore admite el formato de cabecera de rastreo X-Ray para el rastreo distribuido. Puede incluir el `X-Amzn-Trace-Id` encabezado en sus solicitudes para AgentCore mantener el contexto del rastreo más allá de los límites del servicio.

Amazon Elastic Compute Cloud y AWS X-Ray

Puede instalar y ejecutar el daemon de X-Ray en una instancia de Amazon EC2 con un script de datos de usuario. Para obtener instrucciones, consulte Ejecutar el daemon X-Ray en Amazon EC2.

Utilice un perfil de instancia para conceder permiso al daemon para cargar datos de rastro en X-Ray. Para obtener más información, consulte Permiso para el envío de datos a X-Ray desde el daemon.

Amazon SNS y AWS X-Ray

Puede usar AWS X-Ray con Amazon Simple Notification Service (Amazon SNS) para rastrear y analizar las solicitudes a medida que recorren sus temas de SNS hasta sus servicios de suscripción

compatibles con SNS. Utilice el rastreo de X-Ray con Amazon SNS para analizar las latencias de sus mensajes y sus servicios backend, por ejemplo, cuánto tiempo pasa una solicitud en un tema y cuánto tiempo ha tardado en enviar el mensaje a cada una de las suscripciones del tema. Amazon SNS solo admite rastreo de X-Ray para temas estándar y FIFO.

Si publica en un tema de Amazon SNS desde un servicio que ya está instrumentado con X-Ray, Amazon SNS pasa el contexto de rastro del publicador a los suscriptores. Además, puede activar el rastreo activo para enviar datos de segmentos sobre sus suscripciones de Amazon SNS a X-Ray para los mensajes publicados desde un cliente de SNS instrumentado. Active el rastreo activo para un tema de Amazon SNS a través de la consola de Amazon SNS o de la API o la CLI de Amazon SNS. Consulte Instrumentación de su solicitud para obtener más información sobre cómo instrumentar sus clientes de SNS.

Configuración del rastreo activo de Amazon SNS

Puede utilizar la consola de Amazon SNS o la CLI o el SDK de AWS para configurar el rastreo activo de Amazon SNS.

Al utilizar la consola de Amazon SNS, Amazon SNS intenta crear los permisos necesarios para que SNS llame a X-Ray. El intento puede rechazarse si el usuario no tiene los permisos suficientes para modificar las políticas de recursos de X-Ray. Para obtener más información sobre estos permisos consulte Administración de identidades y accesos en Amazon SNS y Ejemplos de casos de control de acceso con Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service. Para obtener más información acerca de cómo activar el rastreo activo a través de la consola de Amazon SNS, consulte Habilitar el rastreo activo en un tema de Amazon SNS en la Guía para desarrolladores de Amazon Simple Notification Service.

Al utilizar la CLI o el SDK de AWS para activar el rastreo activo, debe configurar manualmente los permisos mediante políticas basadas en recursos. Use `PutResourcePolicy` para configurar X-Ray con la política basada en recursos necesaria para permitir que Amazon SNS envíe rastros a X-Ray.

Example Ejemplo de política basada en recursos de X-Ray para el rastreo activo de Amazon SNS

En este ejemplo de documento de política se especifican los permisos que Amazon SNS necesita para enviar datos de rastro a X-Ray:

```
{
    Version: "2012-10-17",
    Statement: [
```

```
      {
        Sid: "SNSAccess",
        Effect: Allow,
        Principal: {
          Service: "sns.amazonaws.com",
        },
        Action: [
          "xray:PutTraceSegments",
          "xray:GetSamplingRules",
          "xray:GetSamplingTargets"
        ],
        Resource: "*",
        Condition: {
          StringEquals: {
            "aws:SourceAccount": "account-id"
          },
          StringLike: {
            "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
          }
        }
      }
    ]
  }
```

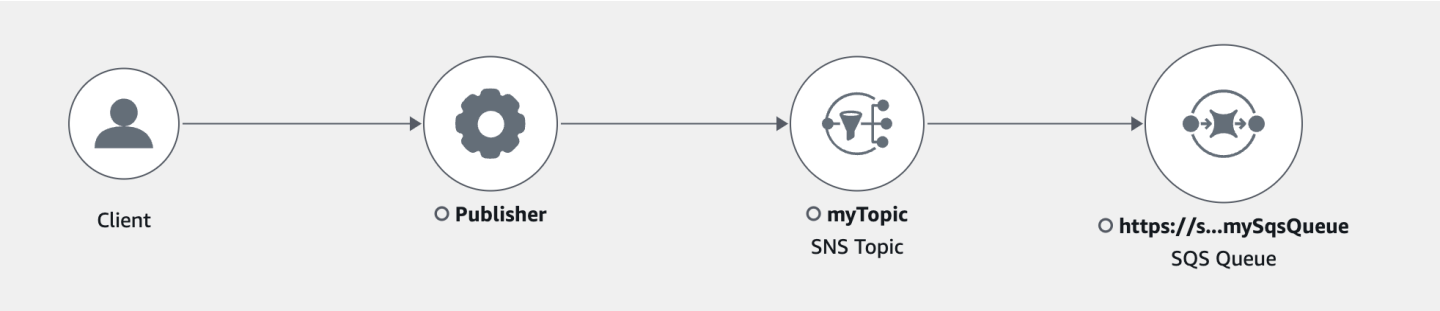
Utilice la CLI para crear una política basada en recursos que dé a Amazon SNS permisos para enviar datos de rastro a X-Ray:

```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
 '{ "Version": "2012-10-17",         "Statement": [ { "Sid": "SNSAccess",
 "Effect": "Allow", "Principal": { "Service": "sns.amazonaws.com" }, "Action":
 [ "xray:PutTraceSegments", "xray:GetSamplingRules", "xray:GetSamplingTargets" ],
 "Resource": "*", "Condition": { "StringEquals": { "aws:SourceAccount": "account-
id" }, "StringLike": { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-
name" } } } ] }'
```

Para usar estos ejemplos, sustituya *partition*, *region*, *account-id* y *topic-name* por su partición de AWS, región, ID de cuenta y nombre de tema de Amazon SNS específicos. Para dar permiso a todos los temas de Amazon SNS para que envíen datos de rastro a X-Ray, sustituya el nombre del tema por *.

Visualización de rastros de publicadores y suscriptores de Amazon SNS en la consola de X-Ray

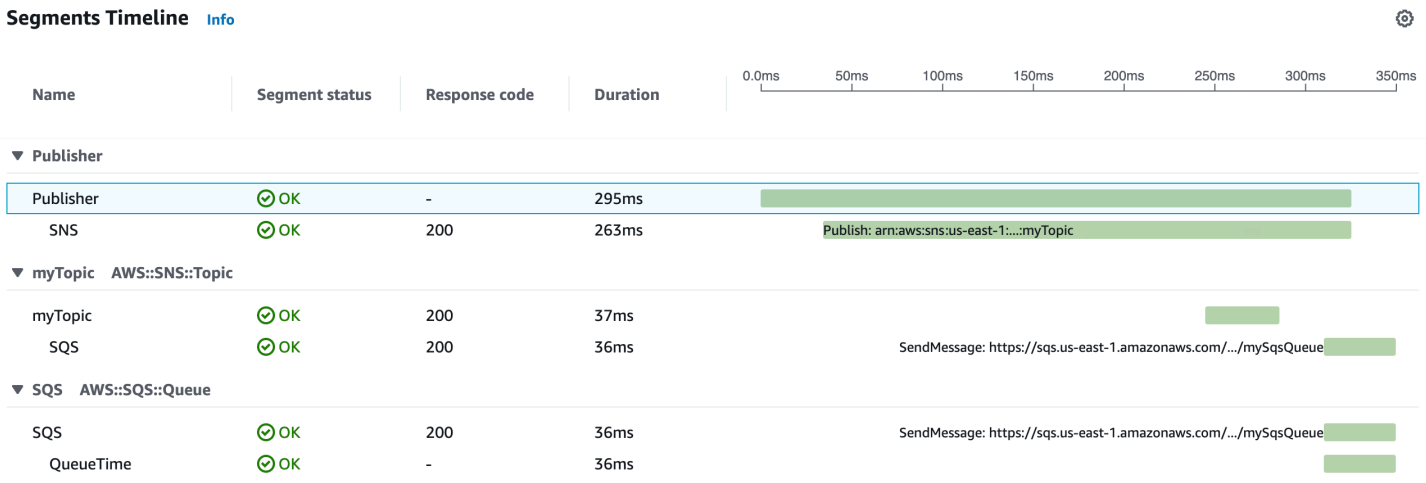
Utilice la consola X-Ray para ver un mapa de rastros y los detalles de rastro que muestran una vista conectada de los publicadores y suscriptores de Amazon SNS. Cuando el rastreo activo de Amazon SNS está activado para un tema, el mapa de rastros y el mapa de detalles de rastro de X-Ray muestran los nodos conectados para los publicadores de Amazon SNS, el tema de Amazon SNS y los suscriptores posteriores:



Tras elegir un rastro que abarque a un publicador y un suscriptor de Amazon SNS, la página de detalles del rastro de X-Ray muestra un mapa de detalles de rastro y una escala de tiempo de segmentos.

Example Ejemplo de escala de tiempo con el publicador y el suscriptor de Amazon SNS

En este ejemplo se muestra una escala de tiempo que incluye a un publicador de Amazon SNS que envía un mensaje a un tema de Amazon SNS que es procesado por un suscriptor de Amazon SQS.

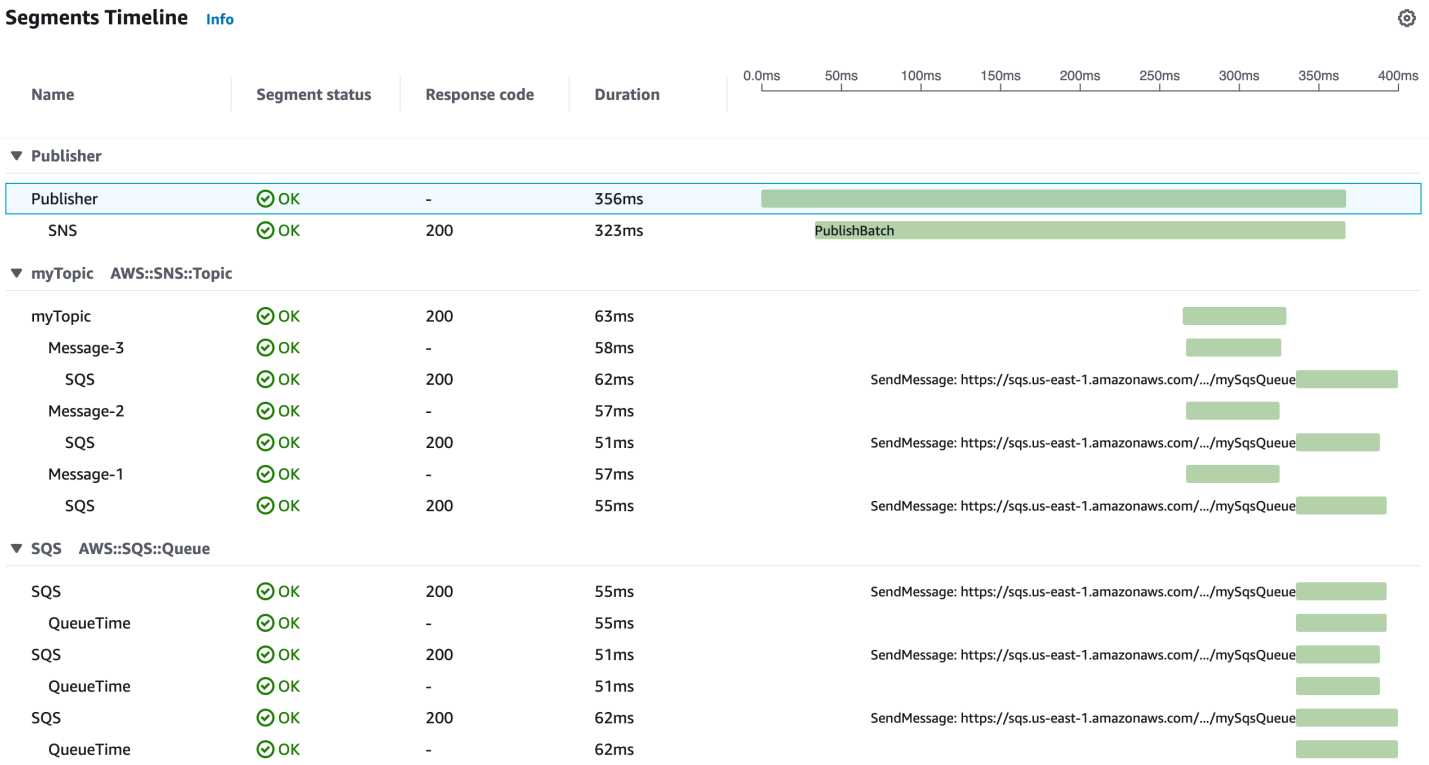


El ejemplo de escala de tiempo anterior proporciona detalles sobre el flujo de mensajes de Amazon SNS:

- El segmento SNS representa la duración de ida y vuelta de la llamada a la API `Publish` desde el cliente.
- El segmento myTopic representa la latencia de la respuesta de Amazon SNS a la solicitud de publicación.
- El subsegmento SQS representa el tiempo de ida y vuelta que tarda Amazon SNS en publicar el mensaje en una cola de Amazon SQS.
- El tiempo entre el segmento MyTopic y el subsegmento SQS representa el tiempo que el mensaje pasa en el sistema Amazon SNS.

Example Ejemplo de escala de tiempo con mensajes de Amazon SNS por lotes

Si se agrupan varios mensajes de Amazon SNS en un solo lote de un rastro, la escala de tiempo de los segmentos muestra los segmentos que representan cada mensaje que se procesa.

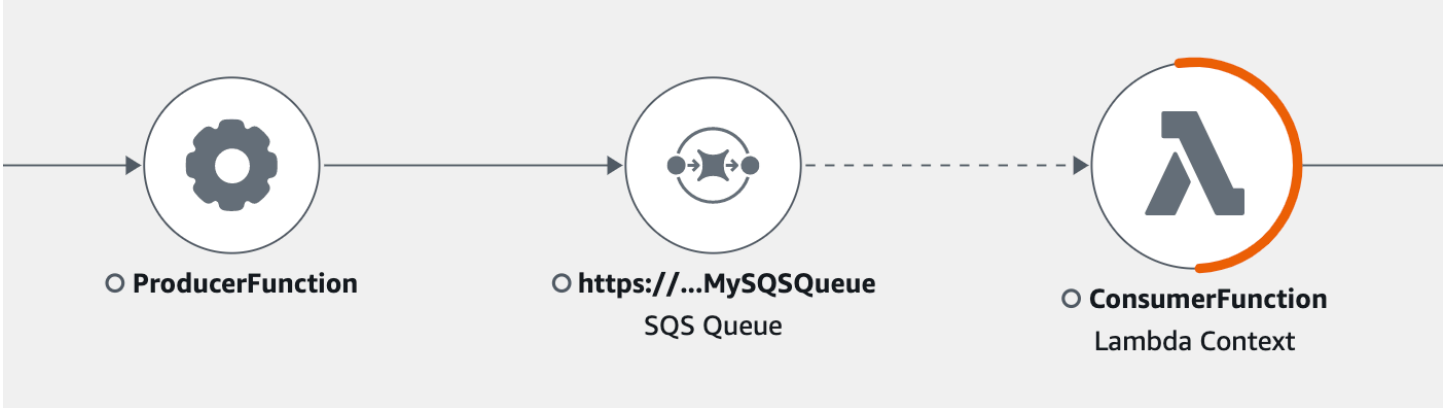


Amazon SQS y AWS X-Ray

AWS X-Ray se integra en Amazon Simple Queue Service (Amazon SQS) para rastrear mensajes que pasan por una cola de Amazon SQS. Si un servicio rastrea solicitudes utilizando el SDK de X-Ray, Amazon SQS podrá enviar el encabezado de rastreo y continuar propagando el rastro original

entre el remitente y el consumidor con un ID de rastro coherente. Esta continuidad permite que los usuarios puedan rastrear, analizar y depurar en otros servicios posteriores.

AWS X-Ray admite el rastreo de aplicaciones basadas en eventos mediante Amazon SQS y AWS Lambda. Utilice la consola de CloudWatch para ver una vista conectada de cada solicitud a medida que se pone en cola con Amazon SQS y la procesa una función de Lambda posterior. Los rastros de los productores de mensajes anteriores se vinculan automáticamente a los rastros de los nodos de consumidor de Lambda posteriores, lo que crea una vista integral de toda la aplicación. Para obtener más información, consulte Rastreo de aplicaciones basadas en eventos.



Amazon SQS admite la siguiente instrumentación de encabezado de rastreo:

- Encabezado HTTP predeterminado: el SDK de X-Ray rellena automáticamente el encabezado de rastro como encabezado HTTP cuando se llama a Amazon SQS a través del SDK de AWS. `X-Amzn-Trace-Id` lleva el encabezado de rastro predeterminado y se corresponde con todos los mensajes incluidos en una solicitud `SendMessage` o `SendMessageBatch`. Para obtener más información sobre el encabezado HTTP predeterminado, consulte Encabezado de seguimiento.
- Atributo del sistema **`AWSTraceHeader`**: `AWSTraceHeader` es un atributo del sistema de mensajes reservado por Amazon SQS para incluir el encabezado de rastro de X-Ray con los mensajes de la cola. `AWSTraceHeader` está disponible para su uso incluso cuando la instrumentación automática a través del SDK de X-Ray no lo está, por ejemplo, al crear un SDK de rastreo para un nuevo lenguaje. Cuando se establecen las instrumentaciones de los dos encabezados, el atributo del sistema de mensajes anula el encabezado de rastreo HTTP.

Cuando se ejecuta en Amazon EC2, Amazon SQS solo permite procesar un mensaje cada vez. Esto es aplicable tanto cuando se ejecuta un host en las instalaciones como cuando se utilizan servicios de contenedor, como AWS Fargate, Amazon ECS o AWS App Mesh.

El encabezado de rastro no se incluye en las cuotas del tamaño de mensaje y de atributo del mensaje de Amazon SQS. Al habilitar el rastreo de X-Ray, no se superan las cuotas de Amazon SQS. Para obtener más información acerca de las cuotas de AWS, consulte Cuotas de Amazon SQS.

Enviar el encabezado de seguimiento HTTP

Los componentes del remitente de Amazon SQS pueden enviar automáticamente el encabezado de rastro utilizando una llamada a `SendMessageBatch` o a `SendMessage`. Cuando se instrumentan clientes del SDK de AWS, se puede realizar un seguimiento automático en todos los lenguajes en el SDK de X-Ray. Los recursos y Servicios de AWS rastreados a los que accede dentro de dichos servicios (por ejemplo, un bucket de Amazon S3 o una cola de Amazon SQS) aparecerán como nodos posteriores en el mapa de rastros en la consola de X-Ray.

Para obtener información acerca de cómo se rastrean las llamadas al SDK de AWS con su lenguaje preferido, consulte los siguientes temas en los SDK correspondientes:

- Go – Rastreo de llamadas AWS del SDK con el X-Ray SDK for Go
- Java: Rastreo de llamadas al AWS SDK con el X-Ray SDK for Java
- Node.js – Rastreo de llamadas al AWS SDK con el SDK de X-Ray para Node.js
- Python – Rastreo de llamadas al AWS SDK con el SDK de X-Ray para Python
- Ruby – Rastreo de llamadas al AWS SDK con el SDK de X-Ray para Ruby
- .NET – Rastreo de llamadas AWS del SDK con el X-Ray SDK para .NET

Recuperar el encabezado y el contexto de seguimiento

Si utiliza un consumidor posterior de Lambda, la propagación del contexto de rastro es automática. Para continuar la propagación del contexto con consumidores de Amazon SQS, debe instrumentar manualmente la entrega al componente receptor.

Hay tres pasos principales para recuperar el contexto de rastreo:

- Recibir el mensaje de la cola del atributo `AWSTraceHeader` llamando a la API `ReceiveMessage`.
- Recuperar el encabezado de rastreo del atributo.
- Recuperar el ID de rastreo del encabezado. Si lo desea, también puede añadir otras métricas al segmento.

A continuación se muestra un ejemplo de una implementación escrita con el SDK de X-Ray para Java.

Example: Recuperar el encabezado y el contexto de seguimiento

```
// Receive the message from the queue, specifying the "AWSTraceHeader"
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest()
        .withQueueUrl(QUEUE_URL)
        .withAttributeNames("AWSTraceHeader");
List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();

if (!messages.isEmpty()) {
    Message message = messages.get(0);

    // Retrieve the trace header from the AWSTraceHeader message system attribute
    String traceHeaderStr = message.getAttributes().get("AWSTraceHeader");
    if (traceHeaderStr != null) {
        TraceHeader traceHeader = TraceHeader.fromString(traceHeaderStr);

        // Recover the trace context from the trace header
        Segment segment = AWSXRay.getCurrentSegment();
        segment.setTraceId(traceHeader.getRootTraceId());
        segment.setParentId(traceHeader.getParentId());

 segment.setSampled(traceHeader.getSampled().equals(TraceHeader.SampleDecision.SAMPLED));
    }
}
```

Amazon S3 y AWS X-Ray

AWS X-Ray se integra en Amazon S3 para rastrear las solicitudes previas para actualizar los buckets de S3 de la aplicación. Si un servicio rastrea las solicitudes mediante el SDK de X-Ray, Amazon S3 puede enviar los encabezados de rastreo a los suscriptores de eventos posteriores, como AWS Lambda, Amazon SQS y Amazon SNS. X-Ray permite rastrear mensajes para las notificaciones de eventos de Amazon S3.

Puede usar el mapa de rastros de X-Ray para ver las conexiones entre Amazon S3 y otros servicios que utiliza su aplicación. También puede utilizar la consola para ver métricas como la latencia media y las tasas de errores. Para obtener más información sobre la consola de X-Ray, consulte Uso de la consola de X-Ray.

Amazon S3 admite la instrumentación de encabezados HTTP predeterminados. El SDK de X-Ray rellena automáticamente el encabezado de rastro como encabezado HTTP cuando se llama a Amazon S3 a través del SDK de AWS. `X-Amzn-Trace-Id` lleva el encabezado de rastro predeterminado. Para obtener más información sobre los encabezados de rastreo, consulte Encabezado de seguimiento en la página de conceptos. La propagación del contexto de rastro de Amazon S3 admite los siguientes suscriptores: Lambda, SQS y SNS. Como SQS y SNS no emiten datos de segmentos por sí mismos, no aparecerán en el rastro o mapa de rastros cuando S3 los active, aunque propagarán el encabezado de rastreo a los servicios posteriores.

Configuración de notificaciones de eventos de Amazon S3

La característica de notificaciones de Amazon S3 le permite recibir notificaciones cuando se producen ciertos eventos en su bucket. A continuación, estas notificaciones se pueden propagar a los siguientes destinos de la aplicación:

- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- AWS Lambda

Para obtener una lista de los eventos compatibles, consulte los tipos de eventos compatibles en la Guía para desarrolladores de Amazon S3.

Amazon SNS y Amazon SQS

Debe conceder permisos a Amazon S3 a fin de publicar notificaciones en un tema de SNS o en una cola de SQS. Para conceder esos permisos debe adjuntar una política de AWS Identity and Access Management (IAM) al tema de SNS o a la cola de SQS de destino. Para obtener más información sobre las políticas de IAM necesarias, consulte Conceder permisos para publicar mensajes en un tema de SNS o en una cola de SQS.

Para obtener información sobre la integración de SNS y SQS en X-Ray, consulte Amazon SNS y AWS X-Ray y Amazon SQS y AWS X-Ray.

AWS Lambda

Cuando utiliza la consola de Amazon S3 para configurar notificaciones de eventos en un bucket de S3 para una función de Lambda, la consola configura los permisos necesarios en la función de Lambda para que Amazon S3 tenga permisos para invocar la función en el bucket. Para obtener más

información, consulte ¿Cómo puedo habilitar y configurar notificaciones de eventos para un bucket de S3? en la Guía del usuario de la consola de Amazon Simple Storage Service.

También puede conceder permisos a Amazon S3 desde AWS Lambda para invocar su función de Lambda. Para obtener más información, consulte Tutorial: Uso de un desencadenador de Amazon S3 para invocar una función de AWS Lambda en la Guía para desarrolladores de AWS Lambda.

Para obtener más información sobre la integración de Lambda en X-Ray, consulte Instrumentación de código Java en AWS Lambda.

AWS Distro para OpenTelemetry y AWS X-Ray

Utilice AWS Distro para OpenTelemetry (ADOT) para recopilar y enviar métricas y rastros a AWS X-Ray y otras soluciones de supervisión, como Amazon CloudWatch, Amazon OpenSearch Service y Amazon Managed Service para Prometheus.

AWS Distro para OpenTelemetry

AWS Distro para OpenTelemetry (ADOT) es una distribución de AWS basada en el proyecto OpenTelemetry de la Cloud Native Computing Foundation (CNCF). OpenTelemetry proporciona un conjunto único de API, bibliotecas y agentes de código abierto para recopilar rastros y métricas distribuidos. Este kit de herramientas es una distribución de componentes originales de OpenTelemetry que incluye SDK, agentes de instrumentación automática y recopiladores probados, optimizados, protegidos y compatibles con AWS.

Con ADOT, los ingenieros pueden instrumentar sus aplicaciones una vez y enviar métricas y rastros correlacionados a varias soluciones de supervisión de AWS, entre ellas Amazon CloudWatch, AWS X-Ray, Amazon OpenSearch Service y Amazon Managed Service para Prometheus.

ADOT se integra en un número cada vez mayor de Servicios de AWS para simplificar el envío de rastros y métricas a soluciones de supervisión como, por ejemplo, X-Ray. Algunos ejemplos de servicios integrados en ADOT:

- AWS Lambda: las capas Lambda administradas por AWS para ADOT permiten que el usuario empiece directamente, al instrumentar automáticamente una función de Lambda y empaquetar OpenTelemetry junto con una configuración lista para usar para AWS Lambda y X-Ray en una capa fácil de configurar. Los usuarios pueden habilitar y deshabilitar OpenTelemetry para su función de Lambda sin cambiar el código. Para obtener más información, consulte AWS Distro para OpenTelemetry Lambda.

- Amazon Elastic Container Service (ECS): recopile métricas y rastros de las aplicaciones de Amazon ECS mediante el AWS Distro para OpenTelemetry Collector, para enviarlos a X-Ray y a otras soluciones de supervisión. Para obtener más información, consulte Collecting application trace data en la Guía del desarrollador de Amazon ECS.
- AWS App Runner: App Runner admite el envío de rastros a X-Ray mediante AWS Distro para OpenTelemetry (ADOT). Utilice los SDK de ADOT para recopilar datos de rastro para sus aplicaciones en contenedores y utilice X-Ray para analizar y obtener información sobre su aplicación instrumentada. Para obtener más información, consulte AWS App Runner and X-Ray.

Para obtener más información sobre AWS Distro para OpenTelemetry, incluida la integración en otros Servicios de AWS, consulte la documentación de AWS Distro para OpenTelemetry.

Para obtener más información sobre cómo instrumentar la aplicación con AWS Distro para OpenTelemetry y X-Ray, consulte Instrumentación de su aplicación con AWS Distro para OpenTelemetry.

Rastreo de los cambios en la configuración de cifrado de X-Ray con AWS Config

AWS X-Ray se integra en AWS Config para registrar cambios de configuración realizados en los recursos de cifrado de X-Ray. Puede utilizar AWS Config para realizar un inventario de recursos de cifrado de X-Ray, auditar el historial de configuración de X-Ray y enviar notificaciones basadas en los cambios de recursos.

AWS Config admite el registro de los siguientes cambios de recursos de cifrado de X-Ray como eventos:

- Cambios de configuración: cambiar o añadir una clave de cifrado o volver a la configuración de cifrado de X-Ray predeterminada.

Utilice las siguientes instrucciones para obtener información sobre cómo crear una conexión básica entre X-Ray y AWS Config.

Creación de un disparador de función de Lambda

Debe tener el ARN de una función de AWS Lambda personalizada antes de poder generar una regla de AWS Config personalizada. Siga estas instrucciones para crear una función básica con Node.js

que devuelve un valor conforme o no conforme de vuelta a AWS Config en función del estado del recurso `XrayEncryptionConfig`.

Para crear una función de Lambda con un disparador de cambio AWS::XrayEncryptionConfig

1. Abra la consola de Lambda. Seleccione Creación de función.
2. Elija Blueprints (Proyectos) y, a continuación, filtre la biblioteca de proyectos para el proyecto config-rule-change-triggered. Haga clic en el nombre del proyecto o bien elija Configure (Configurar) para continuar.
3. Defina los siguientes campos para configurar el proyecto:
   - En Name (Nombre), escriba un nombre.
   - Para Role (Rol), elija Create new role from template(s) (Crear un rol nuevo a partir de las plantillas).
   - Para Role name (Nombre del rol), escriba un nombre.
   - Para Policy templates (Plantillas de política), elija AWS Config Rules permissions (Permisos de reglas de &CC;).
4. Elija Create function (Crear función) para crear y visualizar su función en la consola de AWS Lambda.
5. Edite el código de la función para reemplazar `AWS::EC2::Instance` por `AWS::XrayEncryptionConfig`. También puede actualizar el campo de descripción para reflejar este cambio.

   Código predeterminado

```
    if (configurationItem.resourceType !== 'AWS::EC2::Instance') {
        return 'NOT_APPLICABLE';
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
        return 'NON_COMPLIANT';
```

   Código actualizado

```
    if (configurationItem.resourceType !== 'AWS::XRay::EncryptionConfig') {
        return 'NOT_APPLICABLE';
```

```
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
        return 'NON_COMPLIANT';
```

6. Añada lo siguiente a su rol de ejecución en IAM para acceso a X-Ray. Estos permisos permiten acceso de solo lectura a sus recursos de X-Ray. Si no se proporciona acceso a los recursos apropiados se producirá un mensaje de fuera de ámbito desde AWS Config cuando se evalúa la función de Lambda asociada a la regla.

```
    {
        "Sid": "Stmt1529350291539",
        "Action": [
            "xray:GetEncryptionConfig"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
```

Creación de una regla de AWS Config personalizada para X-Ray

Cuando se crea la función de Lambda, anote el ARN de la función y vaya a la consola de AWS Config para crear la regla personalizada.

Para crear una regla de AWS Config para X-Ray

1. Abra la página Reglas de la consola de AWS Config.
2. Elija Add rule (Añadir una regla) y, a continuación, elija Add custom rule (Añadir una regla personalizada).
3. En ARN de la función de AWS Lambda, inserte el ARN asociado a la función de Lambda que desea utilizar.
4. Elija el tipo de disparador que desea establecer:

   - Cambios de configuración: AWS Config dispara la evaluación cuando cambia la configuración de un recurso que coincida con el ámbito de regla. La evaluación se realiza después de que AWS Config envía una notificación sobre un cambio de elementos de configuración.
   - Periódica: AWS Config ejecuta las evaluaciones de la regla con la frecuencia que se elija (por ejemplo, cada 24 horas).

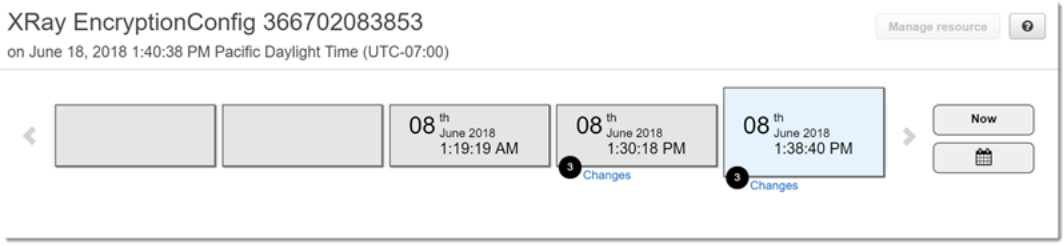
5. Para Tipo de recurso, elija EncryptionConfig en la sección X-Ray.
6. Elija Guardar.

La consola de AWS Config comienza a evaluar la conformidad de la regla de forma inmediata. La evaluación puede tardar varios minutos en completarse.

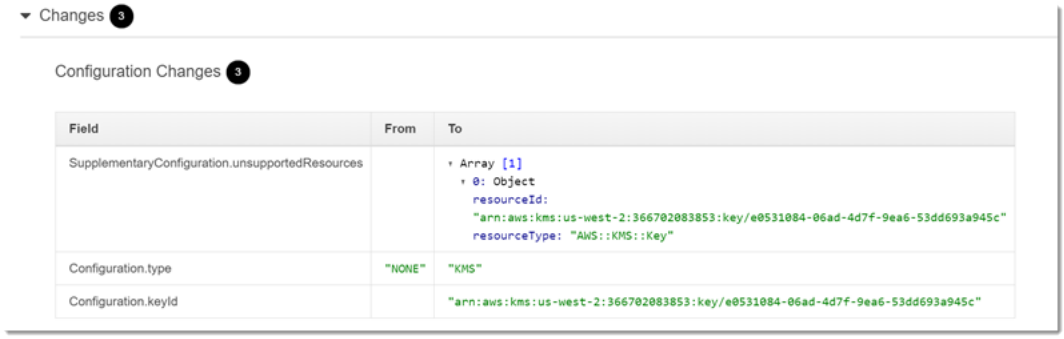
Ahora que esta regla es compatible, AWS Config puede empezar a compilar un historial de auditoría. AWS Config registra los cambios de recursos en forma de escala de tiempo. Para cada cambio en la escala de tiempo de eventos, AWS Config genera una tabla en formato de/a para mostrar lo que ha cambiado en la representación JSON de la clave de cifrado. Los dos cambios de campo asociados a EncryptionConfig son `Configuration.type` y `Configuration.keyID`.

Resultados de ejemplo

A continuación, se muestra un ejemplo de una escala de tiempo de AWS Config que muestra los cambios realizados en fechas y horas concretas.



A continuación, se muestra un ejemplo de una entrada de cambio de AWS Config. El formato de/a ilustra lo que ha cambiado. En este ejemplo se muestra que la configuración de cifrado de X-Ray predeterminada se cambió a una clave de cifrado definida.



Notificaciones de Amazon SNS

Para recibir una notificación de los cambios de configuración, establezca que AWS Config publique las notificaciones de Amazon SNS. Para obtener más información, consulte Monitorización de cambios de recursos de AWS Config por correo electrónico.

AWS AppSync y AWS X-Ray

Puede habilitar y rastrear solicitudes para AWS AppSync. Para obtener más información e instrucciones, consulte Rastreo con AWS X-Ray.

Cuando el seguimiento de X-Ray está habilitado para una API de AWS AppSync, se crea automáticamente un rol vinculado al servicio de AWS Identity and Access Management en su cuenta con los permisos correspondientes. Esto permite a AWS AppSync enviar registros de seguimiento a X-Ray de una manera segura.

Compatibilidad de Amazon API Gateway con el rastreo activo para AWS X-Ray

Puede utilizar X-Ray para rastrear y analizar las solicitudes de los usuarios a medida que viajan a través de las API de Amazon API Gateway a los servicios subyacentes. API Gateway admite el rastreo de X-Ray para todos los tipos de punto de conexión de API Gateway: regional, optimizado para bordes y privado. Puede utilizar X-Ray con Amazon API Gateway en todas las Regiones de AWS donde X-Ray esté disponible. Para obtener más información, consulte Rastro de la ejecución de API de API Gateway conAWS X-Ray en la Guía para desarrolladores de Amazon API Gateway.

> **Note**
>
> X-Ray solo admite el rastreo de las API de REST a través de API Gateway.

Amazon API Gateway admite el rastreo activo para AWS X-Ray. Habilite el rastreo activo en sus etapas de la API para realizar el muestreo de solicitudes entrantes y enviar rastros a X-Ray.

Para habilitar el rastreo activo en una etapa de la API

1. Abra la consola de API Gateway en https://console.aws.amazon.com/apigateway/.
2. Elegir una API.

3. Elegir una etapa.
4. En la pestaña Registros/Rastreo, elija Habilitar el rastreo de X-Ray y, a continuación, seleccione Guardar cambios.
5. Seleccione Resources (Recursos) en el panel de navegación del lado izquierdo.
6. Para volver a implementar la API con la nueva configuración, abra el menú desplegable Acciones y, a continuación, elija Implementar API.

API Gateway utiliza reglas de muestreo que define en la consola de X-Ray para determinar qué solicitudes registrar. Puede crear reglas que solo se apliquen a las API o que solo se apliquen a las solicitudes que contengan determinados encabezados. API Gateway registra los encabezados en atributos del segmento, junto con detalles sobre la etapa y la solicitud. Para obtener más información, consulte Configuración de reglas de muestreo de.

> **Note**
>
> Al rastrear las API de REST con la integración HTTP de API Gateway, el nombre de servicio de cada segmento se establece en la ruta URL de la solicitud desde API Gateway hasta el punto de conexión de la integración HTTP, lo que da como resultado un nodo de servicio en el mapa de rastros de X-Ray para cada ruta URL única. Un gran número de rutas URL puede hacer que el mapa de rastros sobrepase el límite de 10 000 nodos y provocar un error. Para minimizar la cantidad de nodos de servicio creados por API Gateway, considere la posibilidad de pasar los parámetros dentro de la cadena de consulta de URL o en el cuerpo de la solicitud mediante POST. Cualquiera de los dos enfoques garantizará que los parámetros no formen parte de la ruta URL, lo que puede dar lugar a un menor número de rutas URL y nodos de servicio distintos.

Para todas las solicitudes entrantes, API Gateway añade un encabezado de rastreo a las solicitudes HTTP entrantes que no tengan ya uno.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Formato de ID de rastro de X-Ray

Un `trace_id` de X-Ray consta de tres números separados por guiones. Por ejemplo, `1-58406520-a006649127e371903a2de979`. Esto incluye:

- El número de versión, que es 1.
- La hora en que se realizó la solicitud original, en formato de tiempo Unix, en 8 dígitos hexadecimales.

  Por ejemplo, las 10:00 del 1 de diciembre de 2016, PST en formato de tiempo Unix es `1480615200` segundos o `58406520` en formato hexadecimal.
- Un identificador 96 bits del rastro, único a nivel global, en 24 dígitos hexadecimales.

Si el rastreo activo está deshabilitado, la fase sigue registrando un segmento si la solicitud procede de un servicio que ha muestreado la solicitud e iniciado un rastreo. Por ejemplo, una aplicación web instrumentada puede llamar a una API de API Gateway con un cliente HTTP. Cuando instrumente un cliente HTTP con el SDK de X-Ray, este agrega un encabezado de rastreo a la solicitud saliente que contiene la decisión de muestreo. API Gateway lee el encabezado de rastreo y crea un segmento para las solicitudes muestreadas.

Si utiliza API Gateway para generar un SDK de Java para la API, puede instrumentar el cliente del SDK agregando un controlador de solicitudes con el compilador de clientes del mismo modo que instrumentaría manualmente un cliente del SDK de AWS. Para obtener instrucciones, consulte Rastreo de llamadas al AWS SDK con el X-Ray SDK for Java.

Amazon EC2 y AWS App Mesh

AWS X-Ray se integra en AWS App Mesh para administrar los proxies de Envoy para microservicios. App Mesh proporciona una versión de Envoy que puede configurar para enviar datos de rastro al daemon de X-Ray que se ejecute en un contenedor de la misma tarea o pod. X-Ray admite el rastreo con los siguientes servicios compatibles con App Mesh:

- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Amazon Elastic Compute Cloud (Amazon EC2)

Utilice las instrucciones siguientes para aprender a habilitar el seguimiento de X-Ray a través de App Mesh.

Para configurar el proxy de Envoy para enviar datos a X-Ray, establezca la variable de entorno `ENABLE_ENVOY_XRAY_TRACING` en su definición de contenedor.

> **Note**
>
> Actualmente la versión App Mesh de Envoy no envía rastros según las reglas de muestreo configuradas. En su lugar, utiliza un porcentaje de muestreo fijo del 5 % para

la versión 1.16.3 o posterior de Envoy, o un porcentaje de muestreo del 50 % para las versiones de Envoy anteriores a la 1.16.3.

Example Definición del contenedor de Envoy para Amazon ECS

```
{
      "name": "envoy",
      "image": "public.ecr.aws/appmesh/aws-appmesh-envoy:envoy-version",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/myMesh/virtualNode/myNode"
        },
        {
          "name": "ENABLE_ENVOY_XRAY_TRACING",
          "value": "1"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | cut -d' ' -f3 | grep -q live"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      }
```

Note

Para obtener más información sobre las direcciones de región de Envoy disponibles, consulte Imagen de Envoy en la Guía del usuario de AWS App Mesh.

Para obtener más información sobre la ejecución del daemon de X-Ray en un contenedor, consulte Ejecución del daemon de X-Ray en Amazon ECS. Para ver una aplicación de ejemplo que contiene una malla de servicios, un microservicio, un proxy de Envoy y un daemon de X-Ray, implemente el ejemplo `colorapp` del repositorio de GitHub de ejemplos de App Mesh.

Más información

- Introducción a AWS App Mesh
- Introducción a AWS App Mesh y Amazon ECS

AWS App Runner y X-Ray

AWS App Runner es un Servicio de AWS que proporciona una forma rápida, sencilla y rentable de implementar desde el código fuente o una imagen de contenedor directamente hacia una aplicación web escalable y segura en la Nube de AWS. No es necesario aprender nuevas tecnologías, decidir qué servicio informático utilizar o saber cómo aprovisionar y configurar los recursos de AWS. Consulte ¿Qué es AWS App Runner? para obtener más información.

AWS App Runner envía las rastros a X-Ray integrándose en AWS Distro para OpenTelemetry (ADOT). Utilice los SDK de ADOT para recopilar datos de rastro para sus aplicaciones en contenedores y utilice X-Ray para analizar y obtener información sobre su aplicación instrumentada. Para obtener más información, consulte Tracing for your App Runner application with X-Ray.

Registro de llamadas a la API de X-Ray con AWS CloudTrail

AWS X-Ray se integra con AWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un Servicio de AWS. CloudTrail captura todas las llamadas a la API de X-Ray como eventos. Las llamadas capturadas incluyen las que se efectúan desde la consola de X-Ray y las llamadas desde el código a las operaciones de la API de X-Ray. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó en X-Ray, la dirección IP desde la que se realizó, cuándo se realizó y detalles adicionales.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales del usuario raíz o del usuario.
- Si la solicitud se realizó en nombre de un usuario de IAM Identity Center.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro Servicio de AWS.

CloudTrail está activado en la Cuenta de AWS cuando usted crea la cuenta y tiene acceso automático al Historial de eventos de CloudTrail. El Historial de eventos de CloudTrail proporciona un registro visible e inmutable, que se puede buscar y descargar, de los últimos 90 días de eventos de gestión registrados en una Región de AWS. Para obtener más información, consulte Trabajar con el historial de eventos de CloudTrail en la Guía del usuario de AWS CloudTrail. No se cobran cargos de CloudTrail por ver el Historial de eventos.

Para mantener un registro permanente de los eventos en su Cuenta de AWS más allá de los 90 días, cree un registro de seguimiento o un almacén de datos de eventos de CloudTrail Lake.

Registros de seguimiento de CloudTrail

Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. Todos los registros de seguimiento que cree con la Consola de administración de AWS son multirregionales. Puede crear un registro de seguimiento de una sola región o multirregionales mediante la AWS CLI. Se recomienda crear un registro de seguimiento multirregional, ya que registra actividad en todas las Regiones de AWS de su cuenta. Si crea un registro de seguimiento de una sola región, solo podrá ver los eventos registrados en la Región de AWS del registro de seguimiento. Para obtener más información acerca de los registros de seguimiento, consulte Creación de un registro de seguimiento para su Cuenta de AWS y Creación de un registro de seguimiento para una organización en la Guía del usuario de AWS CloudTrail.

Puede crear un registro de seguimiento para enviar una copia de los eventos de administración en curso en su bucket de Amazon S3 sin costo alguno desde CloudTrail; sin embargo, hay cargos por almacenamiento en Amazon S3. Para obtener más información sobre los precios de CloudTrail, consulte Precios de AWS CloudTrail. Para obtener información acerca de los precios de Amazon S3, consulte Precios de Amazon S3.

Almacenes de datos de eventos de CloudTrail Lake

CloudTrail Lake le permite ejecutar consultas basadas en SQL sobre los eventos. CloudTrail Lake convierte los eventos existentes en formato JSON basado en filas al formato ORC de Apache. ORC es un formato de almacenamiento en columnas optimizado para una recuperación rápida de datos. Los eventos se agregan en almacenes de datos de eventos, que son recopilaciones inmutables de eventos en función de criterios que se seleccionan aplicando selectores de eventos avanzados. Los selectores que se aplican a un almacén de datos de eventos controlan los eventos que perduran y están disponibles para la consulta. Para obtener más información acerca de CloudTrail Lake, consulte Trabajar con AWS CloudTrail Lake en la Guía del usuario de AWS CloudTrail.

Los almacenes de datos de eventos de CloudTrail Lake y las consultas generan costos adicionales. Cuando crea un almacén de datos de eventos, debe elegir la opción de precios que desee utilizar para él. La opción de precios determina el costo de la incorporación y el almacenamiento de los eventos, así como el período de retención predeterminado y máximo del almacén de datos de eventos. Para obtener más información sobre los precios de CloudTrail, consulte Precios de AWS CloudTrail.

Temas

Eventos de administración X-Ray en CloudTrail

AWS X-Ray se integra en AWS CloudTrail para registrar las acciones de la API realizadas por un usuario, un rol o un Servicio de AWS en X-Ray. Puede utilizar CloudTrail para supervisar las solicitudes de la API de X-Ray en tiempo real y almacenar registros en Amazon S3, Registros de Amazon CloudWatch y Eventos de Amazon CloudWatch. X-Ray admite el registro de las siguientes acciones como eventos en archivos de registros de CloudTrail:

Acciones de la API admitidas

- PutEncryptionConfig
- GetEncryptionConfig
- CreateGroup
- UpdateGroup
- DeleteGroup
- GetGroup
- GetGroups
- GetInsight
- GetInsightEvents
- GetInsightImpactGraph
- GetInsightSummaries
- GetSamplingStatisticSummaries

Eventos de datos de X-Ray en CloudTrail

Los eventos de datos proporcionan información sobre las operaciones de recursos efectuadas en o dentro de un recurso (por ejemplo, PutTraceSegments, que carga documentos segmentados a X-Ray).

Se denominan también operaciones del plano de datos. Los eventos de datos suelen ser actividades de gran volumen. De forma predeterminada, CloudTrail no registra eventos de datos. El Historial de eventos de CloudTrail no registra los eventos de datos.

Se aplican cargos adicionales a los eventos de datos. Para obtener más información sobre los precios de CloudTrail, consulte Precios de AWS CloudTrail.

Puede registrar eventos de datos para los tipos de recursos de X-Ray mediante la consola de CloudTrail, la AWS CLI o las operaciones de la API de CloudTrail. Para obtener más información sobre cómo registrar los eventos de datos, consulte Registro de eventos de datos con la Consola de administración de AWS y Registro de eventos de datos con la AWS Command Line Interface en la Guía del usuario de AWS CloudTrail.

En la siguiente tabla se muestran los tipos de recursos de X-Ray para los que puede registrar eventos de datos. La columna Tipo de evento de datos (consola) muestra el valor que se debe elegir en la lista de tipos de eventos de datos de la consola de CloudTrail. La columna resources.type value muestra el valor de `resources.type`, que especificaría al configurar los selectores de eventos avanzados mediante la AWS CLI o las API de CloudTrail. La columna API de datos registradas en CloudTrail muestra las llamadas a la API registradas en CloudTrail para el tipo de recurso.

Tipo de evento de datos (consola)	resources.type value	API de datos registradas en CloudTrail
Rastros de X-Ray	`AWS::XRay::Trace`	• PutTraceSegments • GetTraceSummaries • GetTraceGraph • GetServiceGraph • BatchGetTraces • GetTimeSeriesServiceStatistics • PutTelemetryRecords

Tipo de evento de datos (consola)	resources.type value	API de datos registradas en CloudTrail
		• GetSamplingTargets

Puede configurar selectores de eventos avanzados para filtrar según los campos `eventName` y `readOnly` y así registrar solo los eventos que son importantes para usted. Sin embargo, no es posible seleccionar eventos añadiendo el selector de campo `resources.`ARN, ya que los rastros de X-Ray no tienen ARN. Para obtener más información acerca de estos campos, consulte AdvancedFieldSelector en la Referencia de la API de AWS CloudTrail. A continuación, se muestra un ejemplo de cómo ejecutar el comando de la AWS CLI `put-event-selectors` para registrar eventos de datos en un registro de seguimiento de CloudTrail. Debe ejecutar el comando en la región en la que se creó el registro o especificarlo; de lo contrario, la operación devolverá una excepción `InvalidHomeRegionException`.

```
aws cloudtrail put-event-selectors --trail-name myTrail --advanced-event-selectors \
'{
   "AdvancedEventSelectors": [
      {
         "FieldSelectors": [
            { "Field": "eventCategory", "Equals": ["Data"] },
            { "Field": "resources.type", "Equals": ["AWS::XRay::Trace"] },
            { "Field": "eventName", "Equals":
 ["PutTraceSegments","GetSamplingTargets"] }
         ],
         "Name": "Log X-Ray PutTraceSegments and GetSamplingTargets data events"
      }
   ]
}'
```

Ejemplos de eventos de X-Ray

Ejemplos de eventos de administración, **`GetEncryptionConfig`**

En el ejemplo siguiente se muestra una entrada de registro GetEncryptionConfig de X-Ray en CloudTrail.

Example

```
{
```

```
    "eventVersion"=>"1.05",
    "userIdentity"=>{
        "type"=>"AssumedRole",
        "principalId"=>"AROAJVHBZWD3DN6CI2MHM:MyName",
        "arn"=>"arn:aws:sts::123456789012:assumed-role/MyRole/MyName",
        "accountId"=>"123456789012",
        "accessKeyId"=>"AKIAIOSFODNN7EXAMPLE",
        "sessionContext"=>{
            "attributes"=>{
                "mfaAuthenticated"=>"false",
                "creationDate"=>"2023-7-01T00:24:36Z"
            },
            "sessionIssuer"=>{
                "type"=>"Role",
                "principalId"=>"AROAJVHBZWD3DN6CI2MHM",
                "arn"=>"arn:aws:iam::123456789012:role/MyRole",
                "accountId"=>"123456789012",
                "userName"=>"MyRole"
            }
        }
    },
    "eventTime"=>"2023-7-01T00:24:36Z",
    "eventSource"=>"xray.amazonaws.com",
    "eventName"=>"GetEncryptionConfig",
    "awsRegion"=>"us-east-2",
    "sourceIPAddress"=>"33.255.33.255",
    "userAgent"=>"aws-sdk-ruby2/2.11.19 ruby/2.3.1 x86_64-linux",
    "requestParameters"=>nil,
    "responseElements"=>nil,
    "requestID"=>"3fda699a-32e7-4c20-37af-edc2be5acbdb",
    "eventID"=>"039c3d45-6baa-11e3-2f3e-e5a036343c9f",
    "eventType"=>"AwsApiCall",
    "recipientAccountId"=>"123456789012"
}
```

Ejemplos de eventos de datos, **`PutTraceSegments`**

En el ejemplo siguiente se muestra un ejemplo de entrada del registro de eventos PutTraceSegments de X-Ray en CloudTrail.

Example

```
{
```

```
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAWYXPW54Y4NEXAMPLE:i-0dzz2ac111c83zz0z",
    "arn": "arn:aws:sts::012345678910:assumed-role/my-service-role/
i-0dzz2ac111c83zz0z",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAWYXPW54Y4NEXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/service-role/my-service-role",
        "accountId": "012345678910",
        "userName": "my-service-role"
      },
      "attributes": {
        "creationDate": "2024-01-22T17:34:11Z",
        "mfaAuthenticated": "false"
      },
      "ec2RoleDelivery": "2.0"
    }
  },
  "eventTime": "2024-01-22T18:22:05Z",
  "eventSource": "xray.amazonaws.com",
  "eventName": "PutTraceSegments",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.0",
  "userAgent": "aws-sdk-ruby3/3.190.0 md/internal ua/2.0 api/xray#1.0.0 os/linux md/
x86_64 lang/ruby#2.7.8 md/2.7.8 cfg/retry-mode#legacy",
  "requestParameters": {
    "traceSegmentDocuments": [
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0000",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0000",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0001",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0002"
    ]
  },
  "responseElements": {
    "unprocessedTraceSegments": []
  },
  "requestID": "5zzzzz64-acbd-46ff-z544-451a3ebcb2f8",
  "eventID": "4zz51z7z-77f9-44zz-9bd7-6c8327740f2e",
  "readOnly": false,
```

```
  "resources": [
    {
      "type": "AWS::XRay::Trace"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "012345678910",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ZZZZZ-RSA-AAA128-GCM-SHA256",
    "clientProvidedHostHeader": "example.us-west-2.xray.cloudwatch.aws.dev"
  }
}
```

Integración con CloudWatch con X-Ray

AWS X-Ray se integra en CloudWatch Application Signals, CloudWatch RUM y CloudWatch Synthetics para facilitar la supervisión del estado de sus aplicaciones. Habilite su aplicación de Application Signals para que supervise y solucione problemas del estado operativo de sus servicios, páginas de clientes, canarios de Synthetics y dependencias de los servicios.

Al correlacionar las métricas de CloudWatch, los registros y los rastros de X-Ray, el mapa de rastros de X-Ray proporciona una visión integral de sus servicios a fin de ayudarle a detallar rápidamente los cuellos de botella de rendimiento e identificar a los usuarios afectados.

Con CloudWatch RUM, puede realizar una supervisión de usuarios reales para recopilar y ver datos del lado del cliente sobre el rendimiento de su aplicación web desde las sesiones de usuarios reales, casi en tiempo real. Con AWS X-Ray y CloudWatch RUM, puede analizar y depurar la ruta de solicitud empezando por los usuarios finales de su aplicación y pasando por los servicios administrados de AWS posteriores. Eso le ayuda a identificar las tendencias de latencia y los errores que afectan a sus usuarios finales.

Temas

- CloudWatch RUM y AWS X-Ray
- Depuración de valores controlados de CloudWatch Synthetics mediante X-Ray

CloudWatch RUM y AWS X-Ray

Con Amazon CloudWatch RUM, puede realizar una supervisión de usuarios reales para recopilar y ver datos del lado del cliente sobre el rendimiento de su aplicación web desde sesiones de usuarios reales, casi en tiempo real. Con AWS X-Ray y CloudWatch RUM, puede analizar y depurar la ruta de solicitud empezando por los usuarios finales de su aplicación y pasando por los servicios administrados de AWS posteriores. Eso le ayuda a identificar las tendencias de latencia y los errores que afectan a sus usuarios finales.

Una vez activado el rastreo de las sesiones de los usuarios con X-Ray, CloudWatch RUM añade un encabezado de rastro de X-Ray a las solicitudes HTTP permitidas y graba un segmento de X-Ray para las solicitudes HTTP permitidas. A continuación, podrá ver los rastros y segmentos de estas sesiones de usuario en las consolas de X-Ray y CloudWatch, incluido el mapa de rastros de X-Ray.

> **Note**
>
> CloudWatch RUM no se integra en reglas de muestreo de X-Ray. En su lugar, elija un porcentaje de muestreo al configurar la aplicación para que utilice CloudWatch RUM. Los rastros enviados desde CloudWatch RUM pueden conllevar costes adicionales. Para más información, consulte Precios de AWS X-Ray.

Los rastros del lado del cliente enviados desde CloudWatch RUM no están conectados a rastros del lado del servidor de forma predeterminada. Para conectar rastros del lado del cliente con rastros del lado del servidor, configure el cliente web de CloudWatch RUM para agregar un encabezado de rastro de X-Ray a estas solicitudes HTTP.

> **Warning**
>
> La configuración del cliente web de CloudWatch RUM para agregar un encabezado de rastro de X-Ray a las solicitudes HTTP puede ocasionar que el uso compartido de recursos de origen cruzado (CORS) falle. Para evitarlo, añada el encabezado HTTP `X-Amzn-Trace-Id` a la lista de encabezados permitidos en la configuración CORS del servicio posterior. Si utiliza API Gateway como servicio posterior, consulte Habilitar CORS para un recurso de API de REST. Le recomendamos ampliamente que pruebe la aplicación antes de agregar un encabezado de seguimiento de X-Ray del lado del cliente en un entorno de producción. Para obtener más información, consulte la documentación del cliente web de CloudWatch RUM.

Para obtener más información sobre la supervisión de usuarios reales en CloudWatch, consulte Uso de CloudWatch RUM. Para configurar la aplicación para que utilice CloudWatch RUM, incluido el rastreo de las sesiones de los usuarios con X-Ray, consulte Configurar una aplicación para utilizar CloudWatch RUM.

Depuración de valores controlados de CloudWatch Synthetics mediante X-Ray

CloudWatch Synthetics es un servicio completamente administrado que le permite supervisar sus API y puntos de enlace mediante valores controlados con scripts que se ejecutan 24 horas al día, una vez por minuto.

Puede personalizar las scripts de un canary para comprobar si hay cambios en:

- Disponibilidad
- Latencia
- Transacciones
- Vínculos rotos o inactivos
- Finalizaciones de tareas paso a paso
- Errores de carga de páginas
- Latencias de carga para activos de la interfaz de usuario
- Flujos complejos del asistente
- Flujos de compras en la aplicación

Los canaries siguen las mismas rutas y realizan las mismas acciones y comportamientos que sus clientes y verifican de forma continua la experiencia del cliente.

Para obtener más información acerca de la configuración de las pruebas de Synthetics, consulte Uso de Synthetics para crear y administrar canaries.

Los siguientes ejemplos muestran ejemplos comunes de casos de uso para los problemas de depuración que plantean los canaries de Synthetics. Cada ejemplo muestra una estrategia clave para la depuración mediante el mapa de rastros o la consola de X-Ray Analytics.

Para obtener más información acerca de cómo leer e interactuar con el mapa de rastros, consulte Ver el mapa de servicio.

Para obtener más información acerca de cómo leer e interactuar con la consola de X-Ray Analytics, consulte Interactuar con la consola de AWS X-Ray X-Ray Analytics.

Temas

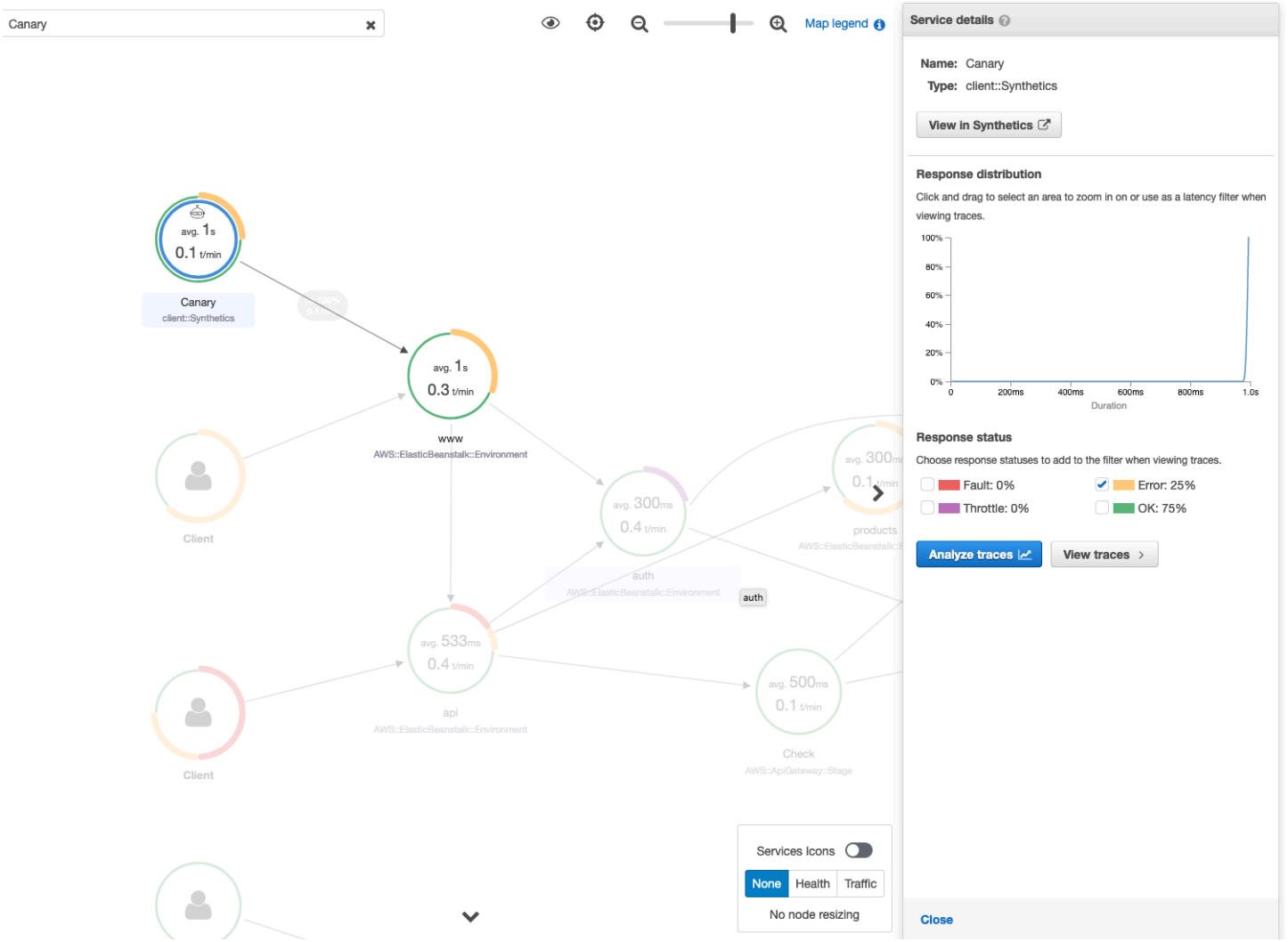
- Consultar canarios con un informe de errores mayor en el mapa de rastros

- Uso de mapas de detalles de rastro para rastros individuales con el fin de consultar cada solicitud con detalle
- Determinar la causa raíz de los errores continuos en los servicios ascendentes y descendentes
- Identificar los cuellos de botella y las tendencias de rendimiento
- Comparar las tasas de error y de latencia antes y después de los cambios
- Determinar la cobertura de canary necesaria para todas las API y URL
- Utilizar los grupos para centrarse en las pruebas de synthetics

Consultar canarios con un informe de errores mayor en el mapa de rastros

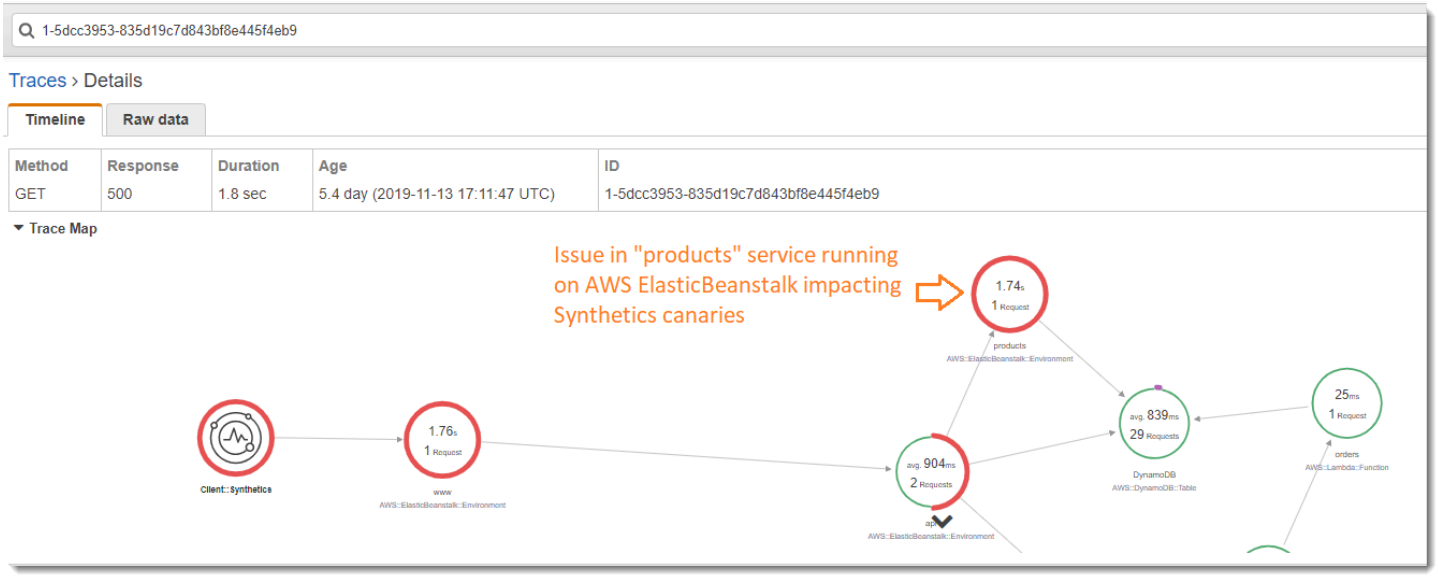
Para ver que canarios tienen un aumento de errores, fallos, tasas de limitación o tiempos de respuesta lentos en su mapa de rastros de X-Ray, puede destacar los nodos de clientes con canarios de Synthetics mediante el filtro `Client::Synthetic`. Al hacer clic en un nodo se muestra la distribución del tiempo de respuesta de toda la solicitud. Al hacer clic en un borde entre dos nodos, se muestran detalles sobre las solicitudes que viajaron por esa conexión. También puede ver los nodos inferidos “remotos” de los servicios posteriores relacionados en su mapa de rastros.

Al hacer clic en el nodo de Synthetics, en el panel lateral aparece el botón Ver en Synthetics, que te redirige a la consola de Synthetics, donde puede comprobar los valores controlados.



Uso de mapas de detalles de rastro para rastros individuales con el fin de consultar cada solicitud con detalle

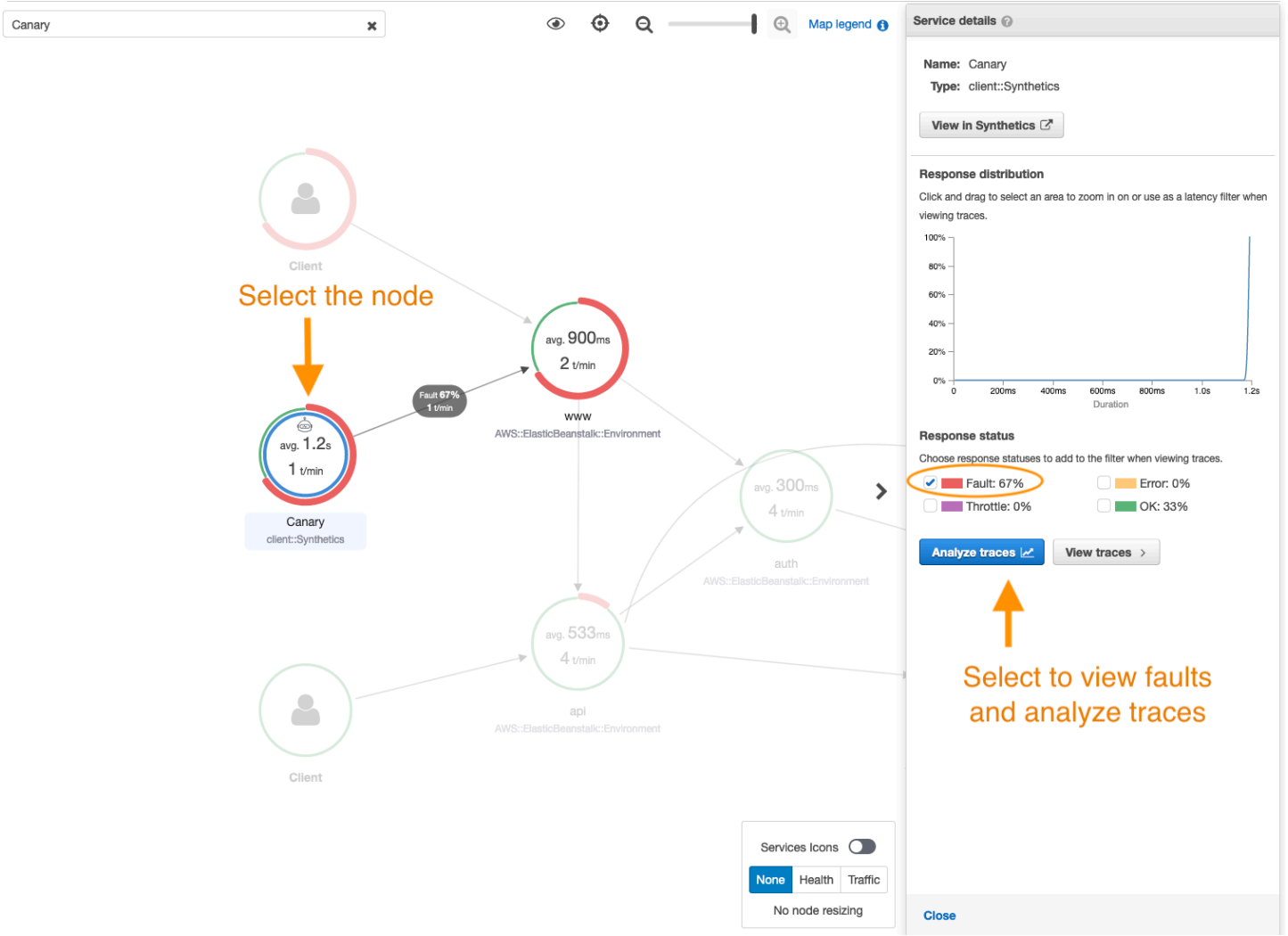
Para determinar qué servicio produce la mayor latencia o que causa un error, invoque el mapa de detalles de rastro al seleccionar el rastreo en el mapa de rastro. Los mapas de detalles de rastro individuales muestran la ruta integral de una sola solicitud. Utilice esto para comprender los servicios invocados y visualizar los servicios ascendentes y descendentes.

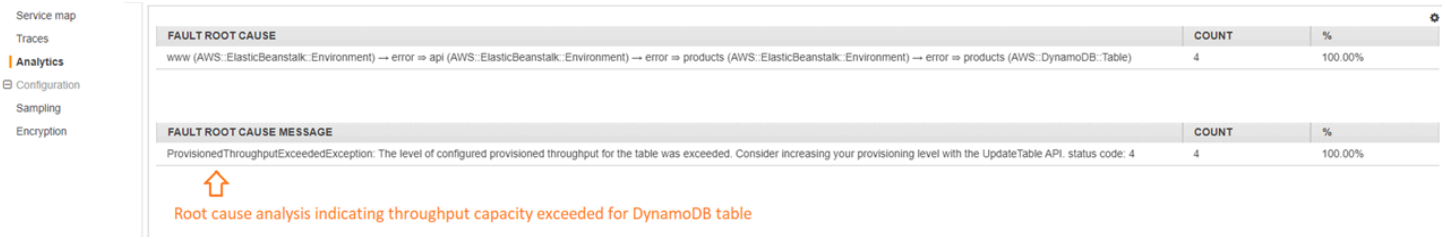


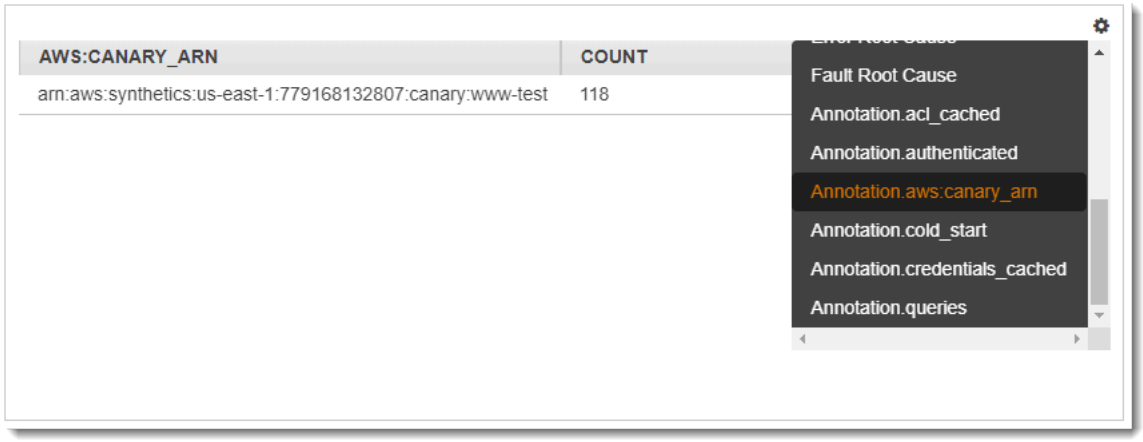
Determinar la causa raíz de los errores continuos en los servicios ascendentes y descendentes

Una vez que reciba una alarma de CloudWatch de fallos en un valor controlado de Synthetics, utilice el modelado estadístico en los datos de rastro en X-Ray para determinar la causa raíz probable del problema en la consola de X-Ray Analytics. En la consola de Analytics, la tabla Causa raíz del tiempo de respuesta muestra las rutas de las entidades registradas. X-ray determina qué ruta del rastro del usuario es la causa más probable del tiempo de respuesta. El formato indica una jerarquía de entidades detectadas, que termina en una causa raíz de tiempo de respuesta.

El siguiente ejemplo muestra que la prueba de Synthetics para la API “XXX” que se ejecuta en la puerta de enlace de la API falla debido a una excepción de capacidad de rendimiento de la tabla de Amazon DynamoDB.

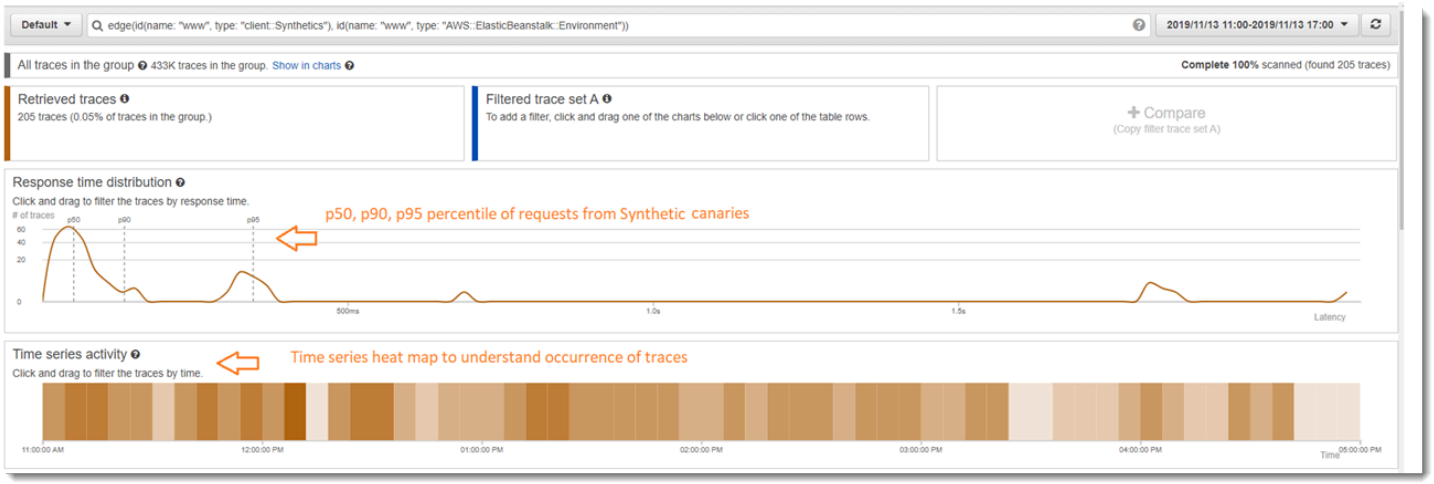

Canary
Map legend
Service details
Name: Canary
Type: client::Synthetics
View in Synthetics
Response distribution
Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.
100%
80%
60%
40%
20%
0%
0
200ms
400ms
600ms
800ms
1.0s
1.2s
Duration
Response status
Choose response statuses to add to the filter when viewing traces.
Fault: 67%
Error: 0%
Throttle: 0%
OK: 33%
Analyze traces
View traces
Select to view faults and analyze traces
Client
Select the node
avg. 900ms
2 t/min
Fault 67%
1 t/min
www
AWS::ElasticBeanstalk::Environment
avg. 1.2s
1 t/min
Canary
client::Synthetics
avg. 300ms
4 t/min
auth
AWS::ElasticBeanstalk::Environment
avg. 533ms
4 t/min
api
AWS::ElasticBeanstalk::Environment
Client
Services Icons
None
Health
Traffic
No node resizing
Close


Service map
Traces
Analytics
Configuration
Sampling
Encryption
FAULT ROOT CAUSE
COUNT
%
www (AWS::ElasticBeanstalk::Environment) ⟶ error ⇒ api (AWS::ElasticBeanstalk::Environment) ⟶ error ⇒ products (AWS::ElasticBeanstalk::Environment) ⟶ error ⇒ products (AWS::DynamoDB::Table)
4
100.00%
FAULT ROOT CAUSE MESSAGE
COUNT
%
ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API. status code: 4
4
100.00%
Root cause analysis indicating throughput capacity exceeded for DynamoDB table



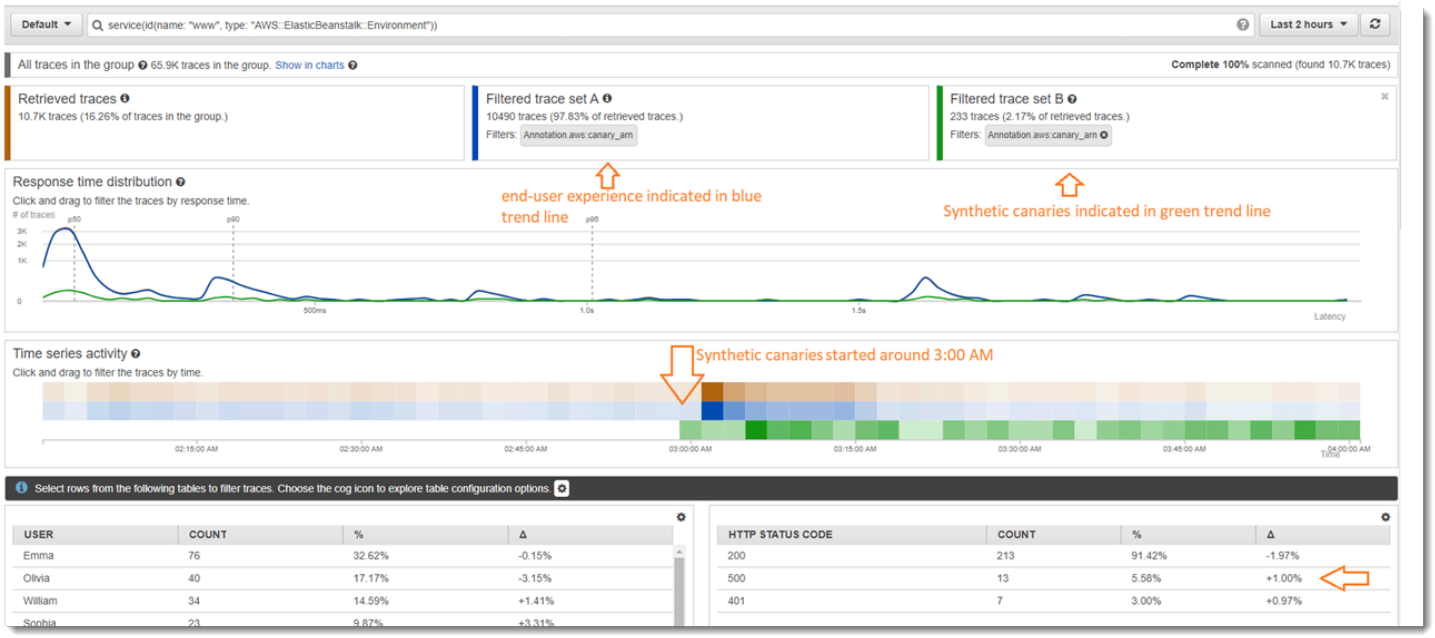
Identificar los cuellos de botella y las tendencias de rendimiento

Puede ver las tendencias del rendimiento de su punto de conexión a lo largo del tiempo mediante el tráfico continuo de sus valores controlados de Synthetics para rellenar un mapa de detalles del rastro durante un período de tiempo.



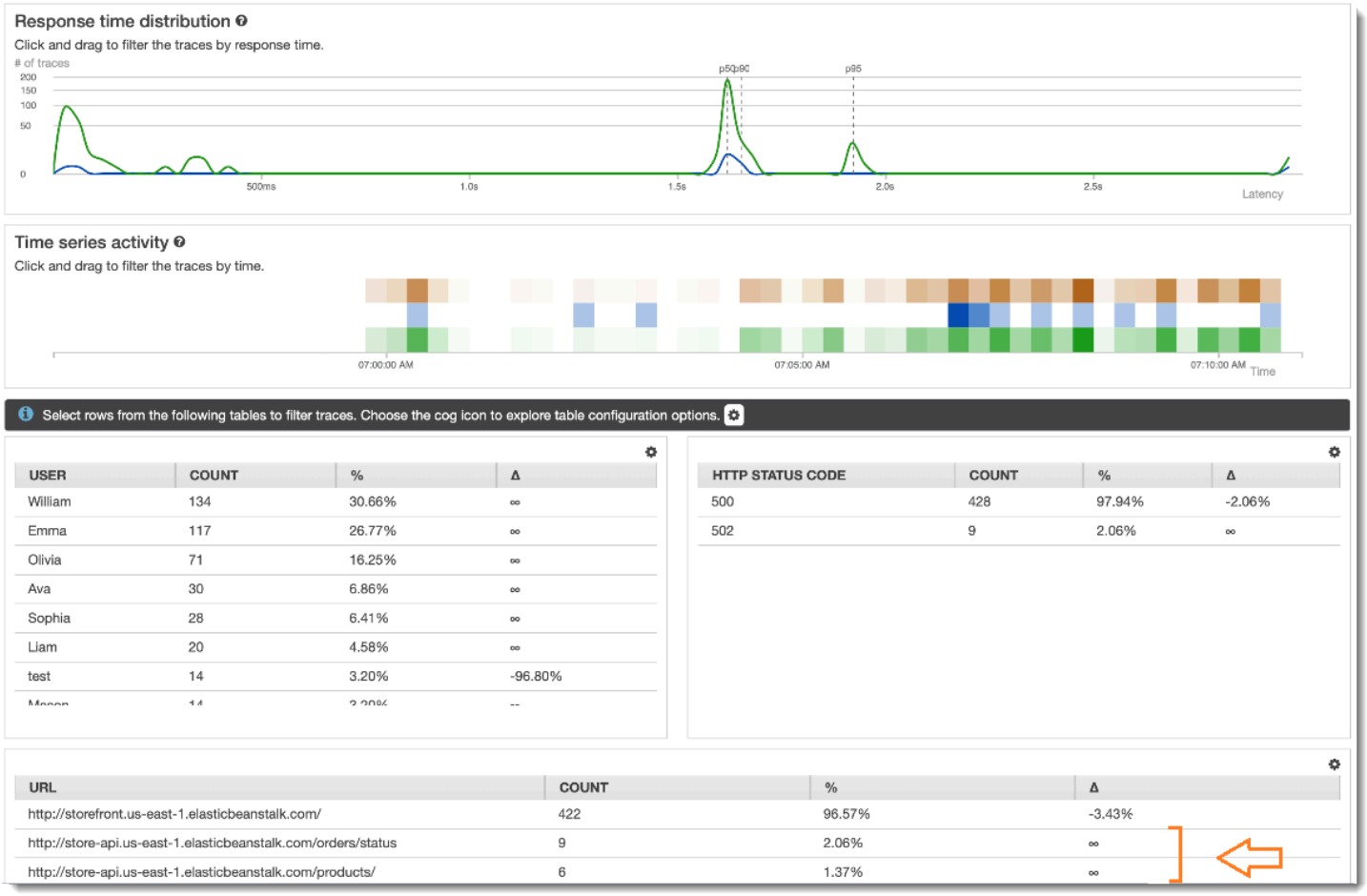
Comparar las tasas de error y de latencia antes y después de los cambios

Señale el momento en el que se produjo un cambio para correlacionar ese cambio a un aumento de los problemas captados por los canarios. Utilice la consola de X-Ray Analytics para definir los intervalos de tiempo anteriores y posteriores como conjuntos de rastros, creando una diferenciación visual en la distribución del tiempo de respuesta.



Determinar la cobertura de canary necesaria para todas las API y URL

Utilice X-Ray Analytics para comparar la experiencia de los canaries con los usuarios. La interfaz de usuario a continuación muestra una línea de tendencia azul para los canaries y una línea verde para los usuarios. También puede identificar las dos URL de tres que no tienen pruebas de canary.

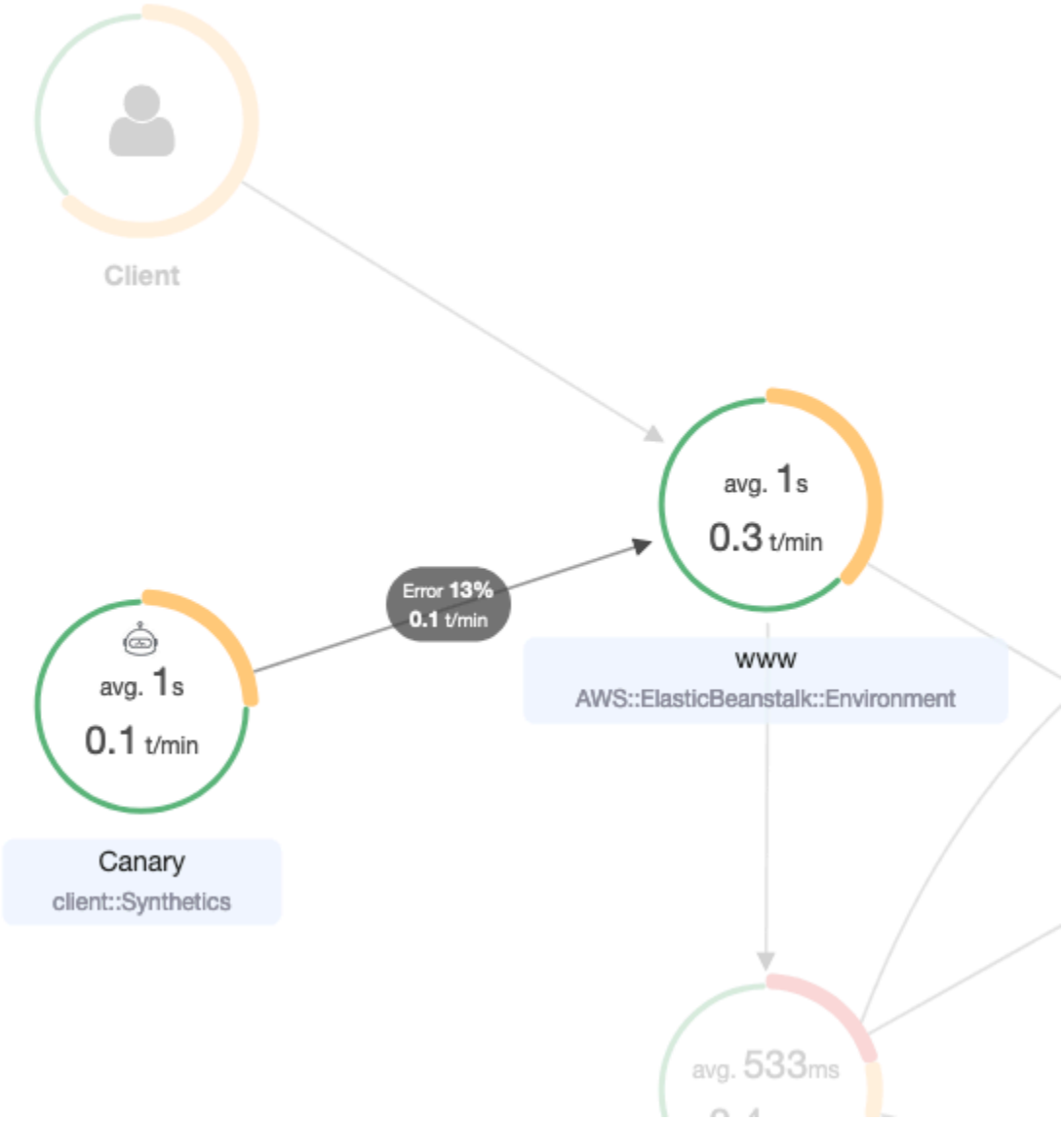


Utilizar los grupos para centrarse en las pruebas de synthetics

Puede crear un grupo de X-Ray utilizando una expresión de filtro para centrarse en un conjunto determinado de flujos de trabajo, como las pruebas de Synthetics para la aplicación “www” que se está ejecutando en AWS Elastic Beanstalk. Utilice las palabras clave complejas `service()` y `edge()` para filtrar por servicios y bordes.

Example Expresión de filtro de grupo

```
"edge(id(name: "www", type: "client::Synthetics"), id(name: "www", type:
 "AWS::ElasticBeanstalk::Environment"))"
```



AWS Elastic Beanstalk y AWS X-Ray

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información

sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

AWS Elastic Beanstalk las plataformas incluyen el daemon X-Ray. Puede ejecutar el daemon mediante la configuración de una opción en la consola de Elastic Beanstalk o con un archivo de configuración.

En la plataforma Java SE, puede utilizar un archivo Buildfile para compilar la aplicación con Maven o Gradle en la instancia. El SDK de X-Ray para Java AWS SDK for Java está disponible en Maven, por lo que puede implementar solo el código de su aplicación y compilarlo a instancia para evitar agrupar y cargar todas sus dependencias.

Puede utilizar las propiedades de entorno de Elastic Beanstalk para configurar el SDK de X-Ray. El método que Elastic Beanstalk utiliza para transferir las propiedades del entorno a su aplicación varía en función de la plataforma. Use las variables de entorno del SDK de X-Ray o las propiedades del sistema según su plataforma.

- Plataforma Node.js: use las variables de entorno.
- Plataforma Java SE: use las variables de entorno.
- Plataforma Tomcat: use las propiedades del sistema.

Para obtener más información, consulta Cómo configurar la AWS X-Ray depuración en la Guía para desarrolladores. AWS Elastic Beanstalk

Elastic Load Balancing y AWS X-Ray

Los equilibradores de carga de aplicación de Elastic Load Balancing añaden un ID de rastro a las solicitudes HTTP entrantes en un encabezado denominado `X-Amzn-Trace-Id`.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Formato de ID de rastro de X-Ray

Un `trace_id` de X-Ray consta de tres números separados por guiones. Por ejemplo, `1-58406520-a006649127e371903a2de979`. Esto incluye:

- El número de versión, que es 1.
- La hora en que se realizó la solicitud original, en formato de tiempo Unix, en 8 dígitos hexadecimales.

  Por ejemplo, las 10:00 del 1 de diciembre de 2016, PST en formato de tiempo Unix es `1480615200` segundos o `58406520` en formato hexadecimal.
- Un identificador 96 bits del rastro, único a nivel global, en 24 dígitos hexadecimales.

Los equilibradores de carga no envían datos a X-Ray ni aparecen como nodo en su mapa de servicio.

Para obtener más información, consulte la sección Rastreo de solicitudes en el equilibrador de carga de aplicaciones en la Guía para desarrolladores de Elastic Load Balancing.

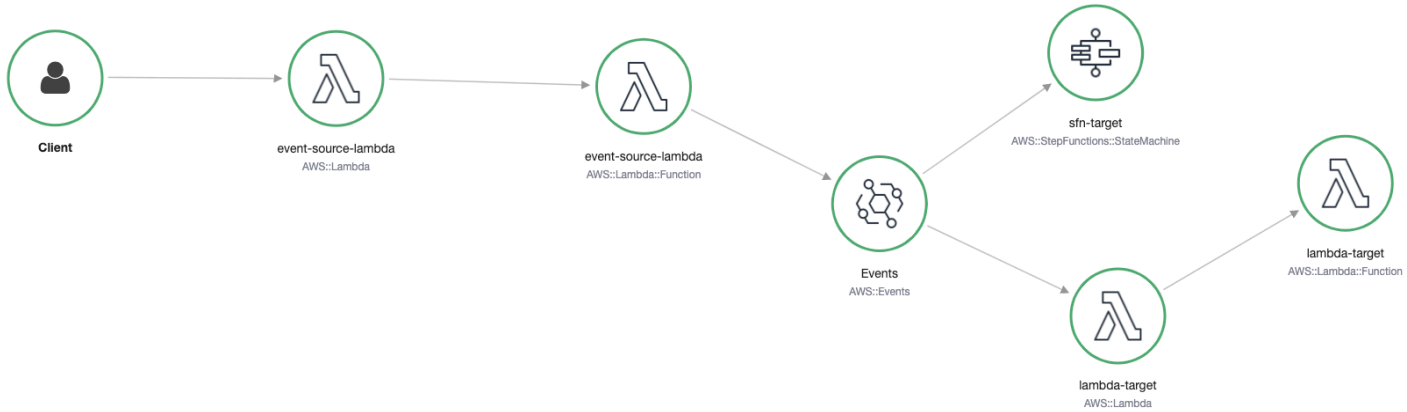
Amazon EventBridge y AWS X-Ray

AWS X-Ray se integra con Amazon EventBridge para rastrear los eventos que se transmiten EventBridge. Si un servicio equipado con el SDK de X-Ray envía eventos a EventBridge, el contexto de rastreo se propaga a los destinos de eventos posteriores dentro del encabezado de rastreo. El SDK de X-Ray recoge automáticamente el encabezado de rastreo y lo aplica a cualquier instrumentación posterior. Esta continuidad permite a los usuarios rastrear, analizar y depurar todos los servicios posteriores y proporciona una visión más completa del sistema.

Para obtener más información, consulte EventBridge X-Ray Integration en la Guía del EventBridge usuario.

Visualización del origen y los destinos en el mapa de servicio de X-Ray

El mapa de rastreo de X-Ray muestra un nodo de EventBridge eventos que conecta los servicios de origen y destino, como en el siguiente ejemplo:



Propague el contexto de rastro a los destinos de eventos.

El SDK de X-Ray permite que la fuente de EventBridge eventos propague el contexto de rastreo a los objetivos de eventos posteriores. Los siguientes ejemplos específicos del idioma muestran la llamada EventBridge desde una función de Lambda en la que está habilitado el rastreo activo:

Java

Agregue las dependencias necesarias para X-Ray:

- AWS X-Ray SDK para Java
- AWS X-Ray SDK de grabadora para Java

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.services.eventbridge.AmazonEventBridge;
import com.amazonaws.services.eventbridge.AmazonEventBridgeClientBuilder;
import com.amazonaws.services.eventbridge.model.PutEventsRequest;
import com.amazonaws.services.eventbridge.model.PutEventsRequestEntry;
import com.amazonaws.services.eventbridge.model.PutEventsResult;
import com.amazonaws.services.eventbridge.model.PutEventsResultEntry;
import com.amazonaws.xray.handlers.TracingHandler;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import java.lang.StringBuilder;
import java.util.Map;
import java.util.List;
import java.util.Date;
import java.util.Collections;

/*
   Add the necessary dependencies for XRay:
   https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-xray
   https://mvnrepository.com/artifact/com.amazonaws/aws-xray-recorder-sdk-aws-sdk
*/
public class Handler implements RequestHandler<SQSEvent, String>{
  private static final Logger logger = LoggerFactory.getLogger(Handler.class);

  /*
    build EventBridge client
  */
  private static final AmazonEventBridge eventsClient =
 AmazonEventBridgeClientBuilder
          .standard()
          // instrument the EventBridge client with the XRay Tracing Handler.
          // the AWSXRay globalRecorder will retrieve the tracing-context
          // from the lambda function and inject it into the HTTP header.
          // be sure to enable 'active tracing' on the lambda function.
          .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
          .build();

  @Override
  public String handleRequest(SQSEvent event, Context context)
  {
    PutEventsRequestEntry putEventsRequestEntry0 = new PutEventsRequestEntry();
    putEventsRequestEntry0.setTime(new Date());
    putEventsRequestEntry0.setSource("my-lambda-function");
    putEventsRequestEntry0.setDetailType("my-lambda-event");
    putEventsRequestEntry0.setDetail("{\"lambda-source\":\"sqs\"}");
    PutEventsRequest putEventsRequest = new PutEventsRequest();
    putEventsRequest.setEntries(Collections.singletonList(putEventsRequestEntry0));
    // send the event(s) to EventBridge
    PutEventsResult putEventsResult = eventsClient.putEvents(putEventsRequest);
    try {
      logger.info("Put Events Result: {}", putEventsResult);
    } catch(Exception e) {
      e.getStackTrace();
```

```
    }
    return "success";
  }
}
```

Python

Agregue la siguiente dependencia a su archivo requirements.txt:

```
aws-xray-sdk==2.4.3
```

```
import boto3
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

# apply the XRay handler to all clients.
patch_all()

client = boto3.client('events')

def lambda_handler(event, context):
    response = client.put_events(
        Entries=[
            {
                'Source': 'foo',
                'DetailType': 'foo',
                'Detail': '{\"foo\": \"foo\"}'
            },
        ]
    )
    return response
```

Go

```
package main

import (
  "context"
  "github.com/aws/aws-lambda-go/lambda"
  "github.com/aws/aws-lambda-go/events"
  "github.com/aws/aws-sdk-go/aws/session"
  "github.com/aws/aws-xray-sdk-go/xray"
  "github.com/aws/aws-sdk-go/service/eventbridge"
```

```
  "fmt"
)

var client = eventbridge.New(session.New())


func main() {
 //Wrap the eventbridge client in the AWS XRay tracer
  xray.AWS(client.Client)
  lambda.Start(handleRequest)
}

func handleRequest(ctx context.Context, event events.SQSEvent) (string, error) {
  _, err := callEventBridge(ctx)
  if err != nil {
    return "ERROR", err
  }
  return "success", nil
}


func callEventBridge(ctx context.Context) (string, error) {
    entries := make([]*eventbridge.PutEventsRequestEntry, 1)
    detail := "{ \"foo\": \"foo\"}"
    detailType := "foo"
    source := "foo"
    entries[0] = &eventbridge.PutEventsRequestEntry{
        Detail: &detail,
        DetailType: &detailType,
        Source: &source,
    }

  input := &eventbridge.PutEventsInput{
     Entries: entries,
  }

  // Example sending a request using the PutEventsRequest method.
  resp, err := client.PutEventsWithContext(ctx, input)

  success := "yes"
  if err == nil { // resp is now filled
      success = "no"
      fmt.Println(resp)
  }
```

```
  return success, err
}
```

Node.js

```
const AWSXRay = require('aws-xray-sdk')
//Wrap the aws-sdk client in the AWS XRay tracer
const AWS = AWSXRay.captureAWS(require('aws-sdk'))
const eventBridge = new AWS.EventBridge()

exports.handler = async (event) => {

  let myDetail = { "name": "Alice" }

  const myEvent = {
    Entries: [{
      Detail: JSON.stringify({ myDetail }),
      DetailType: 'myDetailType',
      Source: 'myApplication',
      Time: new Date
    }]
  }

  // Send to EventBridge
  const result = await eventBridge.putEvents(myEvent).promise()

  // Log the result
  console.log('Result: ', JSON.stringify(result, null, 2))

}
```

C#

Agregue los siguientes paquetes de X-Ray a sus dependencias de C#:

```
<PackageReference Include="AWSXRayRecorder.Core" Version="2.6.2" />
<PackageReference Include="AWSXRayRecorder.Handlers.AwsSdk" Version="2.7.2" />
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Amazon;
```

```
using Amazon.Util;
using Amazon.Lambda;
using Amazon.Lambda.Model;
using Amazon.Lambda.Core;
using Amazon.EventBridge;
using Amazon.EventBridge.Model;
using Amazon.Lambda.SQSEvents;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.AwsSdk;
using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;

[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.Json.JsonSerializer))]

namespace blankCsharp
{
  public class Function
  {
    private static AmazonEventBridgeClient eventClient;

    static Function() {
      initialize();
    }

    static async void initialize() {
      //Wrap the AWS SDK clients in the AWS XRay tracer
      AWSSDKHandler.RegisterXRayForAllServices();
      eventClient = new AmazonEventBridgeClient();
    }

    public async Task<PutEventsResponse> FunctionHandler(SQSEvent invocationEvent,
 ILambdaContext context)
    {
      PutEventsResponse response;
      try
      {
        response = await callEventBridge();
      }
      catch (AmazonLambdaException ex)
      {
        throw ex;
      }
```

```
      return response;
    }

    public static async Task<PutEventsResponse> callEventBridge()
    {
      var request = new PutEventsRequest();
      var entry = new PutEventsRequestEntry();
      entry.DetailType = "foo";
      entry.Source = "foo";
      entry.Detail = "{\"instance_id\":\"A\"}";
      List<PutEventsRequestEntry> entries = new List<PutEventsRequestEntry>();
      entries.Add(entry);
      request.Entries = entries;
      var response = await eventClient.PutEventsAsync(request);
      return response;
    }
  }
}
```

AWS Lambda y AWS X-Ray

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede utilizarla para rastrear sus funciones AWS X-Ray . AWS Lambda Lambada ejecuta el daemon de X-Ray y registra un segmento con detalles sobre la invocación y ejecución de la función. Para realizar una instrumentación adicional, puede agrupar el SDK de X-Ray con su función para registrar las llamadas salientes y añadir anotaciones y metadatos.

Si otro servicio instrumentado llama a la función de Lambda, Lambda rastrea las solicitudes que ya se hayan muestreado sin necesidad de realizar ninguna configuración adicional. El servicio principal

puede ser una aplicación web instrumentada u otra función de Lambda. Su servicio puede invocar la función directamente con un cliente de AWS SDK instrumentado o llamando a una API de API Gateway con un cliente HTTP instrumentado.

AWS X-Ray admite el seguimiento de aplicaciones basadas en eventos mediante Amazon AWS Lambda SQS. Utilice la CloudWatch consola para ver una vista conectada de cada solicitud mientras está en cola con Amazon SQS y procesada por una función Lambda descendente. Las trazas de los productores de mensajes ascendentes se vinculan automáticamente a las trazas de los nodos consumidores de Lambda descendentes, lo que crea una end-to-end vista de la aplicación. Para obtener más información, consulte Rastreo de aplicaciones basadas en eventos.

> **Note**
>
> Si tiene habilitados los rastros para una función de Lambda descendente, también debe tenerlos habilitados para la función de Lambda raíz que llama a la función descendente para que dicha función genere los rastros.

Si un servicio que no está instrumentado invoca la función de Lambda o dicha función se ejecuta en una programación, puede configurar Lambda para que muestree y registre las invocaciones con el rastreo activo.

Para configurar la integración de X-Ray en una AWS Lambda función

1. Abra la consola de AWS Lambda.
2. Seleccione Funciones en el panel de navegación izquierdo.
3. Elija su función.
4. En la pestaña Configuración, desplácese hacia abajo hasta la tarjeta Herramientas de monitorización adicionales. También puede encontrar esta tarjeta seleccionando Herramientas de supervisión y operaciones en el panel de navegación izquierdo.
5. Seleccione Editar.
6. En AWS X-Ray, habilite Rastreo activo.

En los tiempos de ejecución con un SDK de X-Ray correspondiente, Lambda también ejecuta el daemon de X-Ray.

X-Ray SDKs en Lambda

- SDK de X-Ray para Go: tiempos de ejecución de Go 1.7 y versiones más recientes
- SDK de X-Ray para Java: tiempo de ejecución de Java 8
- SDK de X-Ray para Node.js: tiempos de ejecución de Node.js 4.3 y versiones más recientes
- SDK de X-Ray para Python: tiempos de ejecución de Python 2.7, Python 3.6 y versiones más recientes
- SDK de X-Ray para .NET: tiempos de ejecución de .NET Core 2.0 y versiones más recientes

Para utilizar el SDK de X-Ray en Lambda, agrúpelo con el código de la función cada vez que cree una nueva versión. Puede instrumentar las funciones de Lambda con los mismos métodos que utiliza para instrumentar aplicaciones que se ejecutan en otros servicios. La diferencia principal es que no utiliza el SDK para instrumentar las solicitudes entrantes, tomar decisiones de muestreo y crear segmentos.

La otra diferencia entre la instrumentación de funciones de Lambda y aplicaciones web es que el código de su función no puede modificar el segmento que Lambda crea y envía a X-Ray. Puede crear subsegmentos y registrar anotaciones y metadatos en ellos, pero no puede añadir anotaciones y metadatos al segmento principal.

Para obtener más información, consulte Uso de AWS X-Ray en la Guía para desarrolladores de AWS Lambda .

AWS Step Functions y AWS X-Ray

AWS X-Ray se integra en AWS Step Functions para rastrear y analizar las solicitudes de Step Functions. Puede visualizar los componentes de su máquina de estado, identificar cuellos de botella en el rendimiento y solucionar problemas de solicitudes que dieron lugar a un error. Para obtener más información, consulte AWS X-Ray y Step Functions en la Guía para desarrolladores de AWS Step Functions.

Para habilitar el rastreo de X-Ray al crear una nueva máquina de estado

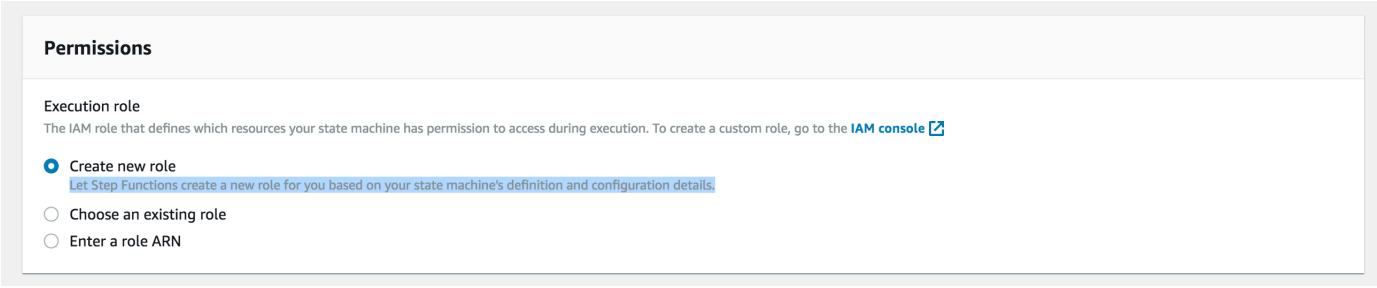
1. Abra la consola de Step Functions en https://console.aws.amazon.com/states/.
2. Elija Crear máquina de estado.
3. En la página Definir una máquina de estado, elija Crear con fragmentos de código o Empezar con una plantilla. Si decide ejecutar un proyecto de muestra, no podrá habilitar el rastreo de

X-Ray durante la creación. En vez de eso, habilite el rastreo de X-Ray después de crear su máquina de estado.

4. Elija Siguiente.
5. En la página Especificar detalles, configure su máquina de estado.
6. Elija Activar rastreo de X-Ray.

Para habilitar el rastreo de X-Ray en una máquina de estado existente

1. En la consola de Step Functions, seleccione la máquina de estado para la que desee activar el rastreo.
2. Elija Editar.
3. Elija Activar rastreo de X-Ray.
4. (Opcional) Genere automáticamente un nuevo rol para su máquina de estado para incluir los permisos de X-Ray seleccionando Crear nuevo rol en la ventana Permisos.



5. Seleccione Save.

> **Note**
>
> Al crear una nueva máquina de estados, se rastrea automáticamente si la solicitud está muestreada y el rastreo está habilitado en un servicio previo como Amazon API Gateway o AWS Lambda. Para cualquier máquina de estado existente que no esté configurada a través de la consola, por ejemplo, a través de una plantilla de CloudFormation, compruebe que tiene una política de IAM que conceda permisos suficientes para habilitar los rastros de X-Ray.

Instrumentación de su solicitud para AWS X-Ray

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

La instrumentación de una aplicación implica el envío de datos de rastro para solicitudes entrantes y salientes y otros eventos de la aplicación junto con los metadatos de cada solicitud. Hay varias opciones de instrumentación entre las que puede elegir o que puede combinar, en función de sus requisitos particulares:

- Instrumentación automática: instrumente su aplicación sin cambios de código, normalmente mediante cambios de configuración, añadiendo un agente de instrumentación automática u otros mecanismos.
- Instrumentación de biblioteca: realice cambios mínimos en el código de la aplicación para añadir instrumentación prediseñada destinada a bibliotecas o marcos específicos, como el AWS SDK, los clientes HTTP de Apache o los clientes SQL.
- Instrumentación manual: añada código de instrumentación a su aplicación en cada ubicación a la que desee enviar la información de rastro.

Existen varios SDKs agentes y herramientas que se pueden utilizar para instrumentar su aplicación de rastreo de rayos X.

Temas

- Cómo equipar su aplicación con la distribución para AWS OpenTelemetry
- Instrumentando su solicitud con AWS X-Ray SDKs
- Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs

Cómo equipar su aplicación con la distribución para AWS OpenTelemetry

The AWS Distro for OpenTelemetry (ADOT) es una AWS distribución basada en el proyecto Cloud Native Computing Foundation (CNCF). OpenTelemetry OpenTelemetry proporciona un conjunto único de código abierto APIs, bibliotecas y agentes para recopilar trazas y métricas distribuidas. Este kit de herramientas es una distribución de OpenTelemetry componentes originales que incluye SDKs agentes de autoinstrumentación y recopiladores que se prueban, optimizan, protegen y respaldan. AWS

Con ADOT, los ingenieros pueden instrumentar sus aplicaciones una vez y enviar métricas y rastreos correlacionados a múltiples soluciones de AWS monitoreo CloudWatch AWS X-Ray, incluidas Amazon y Amazon OpenSearch Service.

El uso de X-Ray con ADOT requiere dos componentes: un OpenTelemetry SDK habilitado para su uso con X-Ray y una AWS Distro for OpenTelemetry Collector habilitada para su uso con X-Ray. Para obtener más información sobre el uso de la AWS distribución OpenTelemetry con AWS X-Ray y otros Servicios de AWS, consulta la documentación de la AWS distribución. OpenTelemetry

Para obtener más información sobre el soporte y el uso de idiomas, consulta AWS Observabilidad en. GitHub

> **Note**
>
> Ahora puedes usar el CloudWatch agente para recopilar métricas, registros y seguimientos de las EC2 instancias de Amazon y de los servidores locales. CloudWatch La versión 1.300025.0 del agente y posteriores puede recopilar rastros de nuestro cliente de OpenTelemetryX-Ray y SDKs enviarlos a X-Ray. Utilizar el CloudWatch agente en lugar del Collector AWS Distro for OpenTelemetry (ADOT) o el daemon X-Ray para recopilar trazas puede ayudarle a reducir la cantidad de agentes que administra. Consulte el tema sobre los CloudWatch agentes en la Guía del CloudWatch usuario para obtener más información.

ADOT incluye lo siguiente:

- AWS Distro for Go OpenTelemetry
- AWS Distro para Java OpenTelemetry
- AWS Distro para OpenTelemetry JavaScript

- AWS Distro para Python OpenTelemetry
- AWS Distro para .NET OpenTelemetry

Actualmente, ADOT admite la instrumentación automática para Java y Python. Además, ADOT permite la instrumentación automática de las funciones de AWS Lambda y sus solicitudes posteriores mediante tiempos de ejecución de Java, Node.js y Python, mediante capas Lambda gestionadas por ADOT.

ADOT SDKs para Java y Go admiten las reglas de muestreo centralizado de X-Ray. Si necesita soporte para las reglas de muestreo de X-Ray en otros idiomas, considere la posibilidad de usar un AWS X-Ray SDK.

> **Note**
>
> Ya puede enviar el rastreo del W3C a IDs X-Ray. De forma predeterminada, las trazas que se crean con ellas OpenTelemetry tienen un formato de ID de traza que se basa en la especificación del contexto de rastreo del W3C. Es diferente del formato de rastreo IDs que se crea mediante un SDK de X-Ray o mediante AWS servicios integrados con X-Ray. Para garantizar que X-Ray acepte IDs el rastreo en formato W3C, debe utilizar la versión 0.86.0 o posterior del AWS X-Ray Exporter, que se incluye en la versión 0.34.0 y posteriores de ADOT Collector. Las versiones anteriores del exportador validan las marcas de tiempo de los identificadores de rastreo, lo que puede provocar que se rechace el rastreo del W3C. IDs

Instrumentando su solicitud con AWS X-Ray SDKs

AWS X-Ray incluye un conjunto de idiomas específicos SDKs para instrumentar su aplicación para enviar trazas a X-Ray. Cada SDK de X-Ray proporciona lo siguiente:

- Interceptadores que añadir a su código para rastrear solicitudes HTTP entrantes
- Controladores de clientes para instrumentar los clientes del AWS SDK que su aplicación utiliza para llamar a otros Servicios de AWS
- Un cliente HTTP para instrumentar llamadas a servicios web HTTP internos y externos

X-Ray SDKs también admite llamadas de instrumentación a bases de datos SQL, instrumentación automática de clientes AWS SDK y otras funciones. En lugar de enviar los datos de rastro

directamente a X-Ray, el SDK envía documentos de segmento JSON a un proceso del daemon que escucha el tráfico UDP. El daemon de X-Ray almacena en búfer segmentos en una cola y los carga en X-Ray en lotes.

Se proporcionan los siguientes idiomas específicos SDKs :

- AWS X-Ray SDK para Go
- AWS X-Ray SDK para Java
- AWS X-Ray SDK para Node.js
- AWS X-Ray SDK para Python
- AWS X-Ray SDK para .NET
- AWS X-Ray SDK para Ruby

Actualmente, X-Ray admite la instrumentación automática para Java.

Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs

Los que SDKs se incluyen con X-Ray son parte de una solución de instrumentación estrechamente integrada que ofrece. AWS El AWS Distro for OpenTelemetry forma parte de una solución industrial más amplia en la que X-Ray es solo una de las muchas soluciones de rastreo. Puede implementar el end-to-end rastreo en X-Ray utilizando cualquiera de los dos enfoques, pero es importante entender las diferencias para determinar cuál es el enfoque más útil para usted.

Le recomendamos que equipe su aplicación con la AWS Distro OpenTelemetry si necesita lo siguiente:

- La posibilidad de enviar rastros a varios backends de rastreo sin tener que volver a instrumentar el código
- Support para una gran cantidad de instrumentaciones bibliotecarias para cada idioma, mantenidas por la comunidad OpenTelemetry
- Capas de Lambda completamente administradas que empaquetan todo lo necesario para recopilar datos de telemetría sin necesidad de cambios de código al usar Java, Python o Node.js

> **Note**
>
> AWS Distro for OpenTelemetry ofrece una experiencia de inicio más sencilla para instrumentar las funciones de Lambda. Sin embargo, debido a la flexibilidad que

OpenTelemetry ofrece, la función Lambda requerirá memoria adicional y las invocaciones pueden experimentar un aumento de la latencia de arranque en frío, lo que puede generar cargos adicionales. Si está optimizando para una baja latencia y no necesita capacidades avanzadas, como destinos OpenTelemetry de back-end configurables de forma dinámica, puede utilizar el SDK de AWS X-Ray para instrumentar su aplicación.

Le recomendamos que elija un SDK de X-Ray para instrumentar su aplicación si necesita lo siguiente:

- Una solución de un solo proveedor perfectamente integrada
- Integración en reglas de muestreo centralizadas de X-Ray, incluida la capacidad de configurar las reglas de muestreo desde la consola de X-Ray y utilizarlas automáticamente en varios hosts, al utilizar Node.js, Python, Ruby o .NET

Transaction Search

Transaction Search es una experiencia de análisis interactiva que puede utilizar para obtener una visibilidad completa de las unidades de seguimiento de transacciones de su aplicación. Las unidades de seguimiento son las unidades de operación fundamentales de un seguimiento distribuido y representan acciones o tareas específicas en una aplicación o un sistema. Cada unidad de seguimiento registra detalles sobre un segmento concreto de la transacción. Estos detalles incluyen las horas de inicio y finalización, la duración y los metadatos asociados, que pueden incluir atributos empresariales como el cliente IDs y el pedido IDs. Las unidades de seguimiento se organizan en una jerarquía principal-secundario. Esta jerarquía forma un rastro completo que asigna el flujo de una transacción entre diferentes componentes o servicios.

Para obtener más información, consulte Transaction Search.

OpenTelemetry Punto final de protocolo (OTLP)

OpenTelemetry es un marco de observabilidad de código abierto que proporciona a los equipos de TI protocolos y herramientas estandarizados para recopilar y enrutar datos de telemetría. Ofrece un formato unificado para instrumentar, generar, recopilar y exportar datos de telemetría de aplicaciones, como métricas, registros y seguimientos, a plataformas de monitoreo para su análisis e información. Al usarlo OpenTelemetry, los equipos pueden evitar la dependencia de un proveedor, lo que garantiza la flexibilidad de sus soluciones de observabilidad.

Puede utilizarlas para enviar rastreos directamente OpenTelemetry a un terminal de OpenTelemetry protocolo (OTLP) y obtener experiencias de supervisión del rendimiento de las aplicaciones de out-of-the Box en Application Signals. CloudWatch

Para obtener más información, consulte OpenTelemetry.

Uso de Go

Puede instrumentar su aplicación Go de dos maneras para enviar rastros a X-Ray:

- AWS Distro para OpenTelemetry Go: una distribución de AWS que proporciona un conjunto de bibliotecas de código abierto para enviar métricas y rastros correlacionados a varias soluciones de supervisión de AWS, incluidas Amazon CloudWatch, AWS X-Ray y Amazon OpenSearch Service, a través de AWS Distro para OpenTelemetry Collector.
- AWS X-Ray SDK for Go: un conjunto de bibliotecas para generar y enviar rastros a X-Ray mediante el daemon X-Ray.

Para obtener más información, consulte Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs.

AWS Distro para Go OpenTelemetry

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Con AWS Distro for OpenTelemetry Go, puede instrumentar sus aplicaciones una vez y enviar métricas y rastreos correlacionados a varias soluciones de AWS monitoreo CloudWatch AWS X-Ray, incluidas Amazon y Amazon OpenSearch Service. El uso de X-Ray con AWS Distro for OpenTelemetry requiere dos componentes: un OpenTelemetry SDK habilitado para su uso con X-Ray y una AWS Distro for OpenTelemetry Collector habilitada para su uso con X-Ray.

Para empezar, consulta la documentación de AWS Distro for OpenTelemetry Go.

Para obtener más información sobre el uso de AWS Distro for OpenTelemetry with AWS X-Ray y otros Servicios de AWS, consulte AWS Distro for OpenTelemetry o Distro for Documentation AWS . OpenTelemetry

Para obtener más información sobre el soporte y el uso de idiomas, consulta AWS Observabilidad en. GitHub

AWS X-Ray SDK for Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Go es un conjunto de bibliotecas para aplicaciones Go que proporcionan clases y métodos para generar y enviar datos de rastro al daemon de X-Ray. Los datos de rastreo incluyen información sobre las solicitudes HTTP entrantes atendidas por la aplicación y las llamadas que la aplicación realiza a los servicios descendentes mediante el AWS SDK, los clientes HTTP o un conector de base de datos SQL. También puede crear segmentos de forma manual y agregar información de depuración en anotaciones y metadatos.

Descarga el SDK desde su GitHubrepositorio congo `get`:

```
$ go get -u github.com/aws/aws-xray-sdk-go/...
```

Para las aplicaciones web, comience por utilizar la función `xray.Handler` para realizar el seguimiento de las solicitudes entrantes. El controlador de mensajes crea un segmento para cada solicitud rastreada y lo completa cuando se envía la respuesta. Mientras el segmento está abierto, puede utilizar los métodos del cliente del SDK para añadir información al segmento y crear subsegmentos para rastrear llamadas posteriores. El SDK también registra automáticamente las excepciones que produce su aplicación mientras el segmento está abierto.

En el caso de las funciones de Lambda llamadas por una aplicación o un servicio instrumentados, Lambda lee el encabezado de rastreo y rastrea automáticamente las solicitudes muestreadas. Para otras funciones, puede configurar Lambda con el fin de muestrear y rastrear las solicitudes entrantes. En cualquier caso, Lambda crea el segmento y se lo proporciona al SDK de X-Ray.

> **Note**
>
> En Lambda, el SDK de X-Ray es opcional. Si no lo usa en su función, el mapa de servicio seguirá incluyendo un nodo para el servicio de Lambda y uno para cada función de Lambda. Al añadir el SDK, puede instrumentar el código de función para añadir subsegmentos al segmento de función registrado por Lambda. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

A continuación, envuelva su cliente con una llamada a la función `AWS`. Este paso garantiza que X-Ray instrumente las llamadas a los métodos del cliente. También puede instrumentar llamadas a bases de datos SQL.

En cuanto empiece a utilizar el SDK, personalice su comportamiento configurando la grabadora y el controlador y el middleware. Puede añadir complementos para registrar los datos sobre los recursos informáticos que ejecutan su aplicación, personalizar el comportamiento de muestreo mediante la definición de reglas de muestreo y definir el nivel de log para ver más o menos información del SDK en los logs de las aplicaciones.

Registre información adicional acerca de las solicitudes y el trabajo que la aplicación realiza en anotaciones y metadatos. Las anotaciones son pares sencillos de clave-valor que se indexan para su uso con expresiones de filtro para poder buscar rastros que contengan datos específicos. Las entradas de metadatos son menos restrictivas y pueden registrar objetos y matrices completos, es decir, todo lo que se pueda serializar en JSON.

> **Anotaciones y metadatos**
>
> Las anotaciones y los metadatos son texto arbitrario que se agrega a los segmentos con el SDK de X-Ray. Las anotaciones se indexan para su uso con expresiones de filtro. Los metadatos no se indexan pero se pueden ver en el segmento sin procesar con la consola o la API de X-Ray. Cualquier persona a la que conceda acceso de lectura a X-Ray puede ver estos datos.

Cuando tenga muchos clientes instrumentados en su código, un único segmento de solicitud puede contener un gran número de subsegmentos, uno para cada llamada realizada con un cliente instrumentado. Puede organizar y agrupar los subsegmentos incluyendo las llamadas del cliente en subsegmentos personalizados. Puede crear un subsegmento personalizado para una función completa o para cualquier sección de código, y registrar los metadatos y las anotaciones en el subsegmento en lugar de escribirlo todo en el segmento principal.

Requisitos

El SDK de X-Ray para Go requiere Go 1.9 o una versión posterior.

El SDK depende de las siguientes bibliotecas durante la compilación y el tiempo de ejecución:

- AWS SDK for Go versión 1.10.0 o posterior

Estas dependencias se han declarado en el archivo `README.md` del SDK.

Documentación de referencia

Una vez que haya descargado el SDK, compile y aloje la documentación localmente para verla en un navegador web.

Para ver la documentación de referencia

1. Vaya al directorio `$GOPATH/src/github.com/aws/aws-xray-sdk-go` (Linux o Mac) o la carpeta `%GOPATH%\src\github.com\aws\aws-xray-sdk-go` (Windows)
2. Ejecute el comando `godoc`.

   ```
   $ godoc -http=:6060
   ```

3. Abra un navegador en `http://localhost:6060/pkg/github.com/aws/aws-xray-sdk-go/`.

Configuración del SDK de X-Ray para Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de

X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede especificar la configuración del SDK de X-Ray para Go a través de variables de entorno, llamando a `Configure` con un objeto `Config` o suponiendo valores predeterminados. Las variables de entorno tienen prioridad sobre los valores `Config`, que tienen prioridad sobre cualquier valor predeterminado.

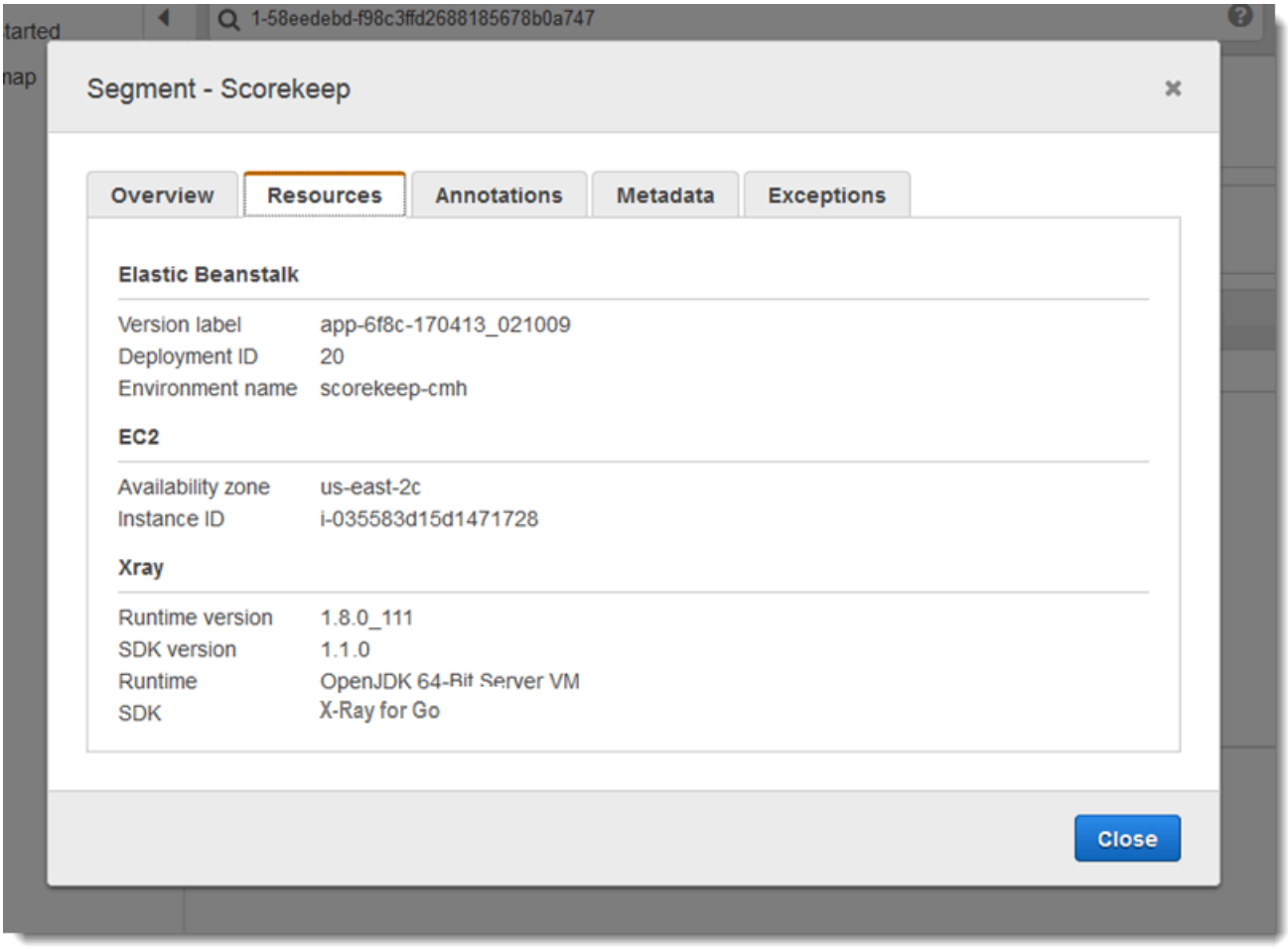
Secciones

Complementos del servicio

Utilice `plugins` para registrar información sobre el servicio que aloja la aplicación.

Plugins

- Amazon EC2 : `EC2Plugin` agrega el ID de la instancia, la zona de disponibilidad y el grupo de CloudWatch registros.
- Elastic Beanstalk: `ElasticBeanstalkPlugin` añade el nombre de entorno, la etiqueta de versión y el ID de implementación.
- Amazon ECS: `ECSPlugin` agrega el ID de contenedor.



Para utilizar un complemento, importe uno de los siguientes paquetes.

```
"github.com/aws/aws-xray-sdk-go/awsplugins/ec2"
"github.com/aws/aws-xray-sdk-go/awsplugins/ecs"
"github.com/aws/aws-xray-sdk-go/awsplugins/beanstalk"
```

Cada complemento tiene una llamada de función `Init()` explícita que lo carga.

Example ec2.Init()

```
import (
 "os"

 "github.com/aws/aws-xray-sdk-go/awsplugins/ec2"
 "github.com/aws/aws-xray-sdk-go/xray"
)

func init() {
```

```
  // conditionally load plugin
  if os.Getenv("ENVIRONMENT") == "production" {
    ec2.Init()
  }

  xray.Configure(xray.Config{
    ServiceVersion: "1.2.3",
  })
}
```

El SDK también usa la configuración del complemento para establecer el campo `origin` en el segmento. Esto indica el tipo de AWS recurso que ejecuta la aplicación. Cuando utilizas varios complementos, el SDK utiliza el siguiente orden de resolución para determinar el origen: ElasticBeanstalk > EKS > ECS > EC2.

Reglas de muestreo

El SDK utiliza las reglas de muestreo que define el usuario en la consola de X-Ray para determinar qué solicitudes registrar. La regla predeterminada rastrea la primera solicitud cada segundo y el 5 % de las solicitudes adicionales de todos los servicios que envían rastros a X-Ray. Cree reglas adicionales en la consola de X-Ray para personalizar la cantidad de datos registrados para cada una de sus aplicaciones.

El SDK aplica las reglas personalizadas en el orden en que se definen. Si una solicitud coincide con varias reglas personalizadas, el SDK solo aplica la primera regla.

> **Note**
>
> Si el SDK no puede comunicarse con X-Ray para obtener las reglas de muestreo, vuelve a una regla local predeterminada de la primera solicitud cada segundo y del 5 % de las solicitudes adicionales por host. Esto puede ocurrir si el host no tiene permiso para llamar al muestreo APIs o no puede conectarse al daemon X-Ray, que actúa como proxy TCP para las llamadas a la API realizadas por el SDK.

También puede configurar el SDK para que cargue las reglas de muestreo desde un documento JSON. El SDK puede usar las reglas locales como respaldo para los casos en que el muestreo de X-Ray no esté disponible, o puede usar las reglas locales exclusivamente.

Example sampling-rules.json

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

En este ejemplo se define una regla personalizada y una regla predeterminada. La regla personalizada aplica un porcentaje de muestreo del 5 % sin un número mínimo de solicitudes de rastreo para las rutas incluidas bajo `/api/move/`. La regla predeterminada rastrea la primera solicitud cada segundo y el 10 % de las solicitudes adicionales.

La desventaja de definir las reglas localmente es que el objetivo establecido lo aplica cada instancia de la grabadora de forma independiente, en lugar de ser administrado por el servicio de X-Ray. A medida que se implementan más hosts, el porcentaje establecido se multiplica, lo que dificulta el control de la cantidad de datos registrados.

Sí AWS Lambda, no puedes modificar la frecuencia de muestreo. Si un servicio instrumentado llama a su función, Lambda registrará las llamadas que generaron solicitudes muestreadas por ese servicio. Si el rastreo activo está activado y no hay ningún encabezado de rastreo, Lambda toma la decisión de muestreo.

Para proporcionar reglas de copia de seguridad, apunte al archivo JSON de muestreo local mediante `NewCentralizedStrategyWithFilePath`.

Example main.go: regla de muestreo local

```
s, _ := sampling.NewCentralizedStrategyWithFilePath("sampling.json") // path to local
 sampling json
```

```
xray.Configure(xray.Config{SamplingStrategy: s})
```

Para utilizar solo reglas locales, apunte al archivo JSON de muestreo local mediante `NewLocalizedStrategyFromFilePath`.

Example main.go — Deshabilitar el muestreo

```
s, _ := sampling.NewLocalizedStrategyFromFilePath("sampling.json") // path to local
 sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Registro

> **Note**
>
> Los campos `xray.Config{}` `LogLevel` y `LogFormat` están obsoletos desde la versión 1.0.0-rc.10.

X-Ray utiliza la siguiente interfaz para el registro. El registrador predeterminado escribe en `stdout` a `LogLevelInfo` y superior.

```
type Logger interface {
 Log(level LogLevel, msg fmt.Stringer)
}

const (
 LogLevelDebug LogLevel = iota + 1
 LogLevelInfo
 LogLevelWarn
 LogLevelError
)
```

Example escribir en **`io.Writer`**

```
xray.SetLogger(xraylog.NewDefaultLogger(os.Stderr, xraylog.LogLevelError))
```

Variables de entorno

Puede usar variables de entorno con el fin de configurar el SDK de X-Ray para Go. El SDK admite las siguientes variables.

- `AWS_XRAY_CONTEXT_MISSING`: establezca esta opción en `RUNTIME_ERROR` para generar excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.

  Valores válidos

  - `RUNTIME_ERROR`: lance una excepción de tiempo de ejecución.
  - `LOG_ERROR`: registre un error y continúe (predeterminado).
  - `IGNORE_ERROR`: ignore el error y continúe.

  Se pueden producir errores relativos a segmentos o subsegmentos inexistentes al intentar usar un cliente instrumentado en el código de inicio que se ejecuta cuando no hay ninguna solicitud abierta o en el código que inicia un nuevo subproceso.
- `AWS_XRAY_TRACING_NAME`: establezca el nombre de servicio que el SDK utiliza para los segmentos.
- `AWS_XRAY_DAEMON_ADDRESS`: establezca el host y el puerto del oyente del daemon de X-Ray. De forma predeterminada, el SDK envía los datos de rastreo a `127.0.0.1:2000`. Use esta variable si ha configurado el daemon para que escuche en un puerto diferente o si se ejecuta en un host diferente.
- `AWS_XRAY_CONTEXT_MISSING`: establezca el valor para determinar cómo administra el SDK los errores de falta de contexto. Se pueden producir errores relativos a segmentos o subsegmentos inexistentes al intentar usar un cliente instrumentado en el código de inicio que se ejecuta cuando no hay ninguna solicitud abierta o en el código que inicia un nuevo subproceso.
  - `RUNTIME_ERROR`: de forma predeterminada, el SDK está configurado para generar una excepción de tiempo de ejecución.
  - `LOG_ERROR`: establezca esta variable para registrar un error y continuar.

Las variables de entorno anulan los valores equivalentes establecidos en el código.

Uso de la opción configure

También puede configurar el SDK de X-Ray para Go con el método `Configure`. `Configure` toma un argumento, un objeto `Config`, con los siguientes campos opcionales.

DaemonAddr

Esta cadena especifica el host y el puerto del oyente del daemon de X-Ray. Si no se especifica, X-Ray utiliza el valor de la variable de entorno `AWS_XRAY_DAEMON_ADDRESS`. Si dicho valor no está establecido, utiliza "127.0.0.1:2000".

ServiceVersion

Esta cadena especifica la versión del servicio. Si no se especifica, X-Ray utiliza la cadena vacía (“”).

SamplingStrategy

Este objeto `SamplingStrategy` especifica las llamadas de aplicación a las que se realiza seguimiento. Si no se especifica, X-Ray utiliza una `LocalizedSamplingStrategy`, que adopta la estrategia tal como se define en `xray/resources/DefaultSamplingRules.json`.

StreamingStrategy

Este `StreamingStrategy` objeto especifica si se debe transmitir un segmento cuando RequiresStreamingdevuelve el valor verdadero. Si no se especifica, X-Ray utiliza un `DefaultStreamingStrategy` que transmite un segmento muestreado si el número de subsegmentos es mayor que 20.

ExceptionFormattingStrategy

Este objeto `ExceptionFormattingStrategy` especifica la forma en que desea gestionar diversas excepciones. Si no se especifica, X-Ray utiliza una `DefaultExceptionFormattingStrategy` con un `XrayError` de tipo `error`, el mensaje de error y rastro de la pila.

Instrumentación de las solicitudes HTTP entrantes con el SDK de X-Ray para Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede usar el SDK de X-Ray para rastrear las solicitudes HTTP entrantes que su aplicación sirve en una EC2 instancia de Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Utilice `xray.Handler` para instrumentar las solicitudes HTTP entrantes. El SDK de X-Ray para Go implementa la interfaz `http.Handler` de la biblioteca de Go estándar en la clase `xray.Handler` para interceptar solicitudes web. La clase `xray.Handler` envuelve el `http.Handler` proporcionado con `xray.Capture` utilizando el contexto de la solicitud, analizando los encabezados entrantes, añadiendo encabezados de respuesta si es necesario y, además, establece los campos de rastreo específicos de HTTP.

Al utilizar esta clase para gestionar solicitudes HTTP y respuestas, el SDK de X-Ray para Go crea un segmento para cada solicitud muestreada. Este segmento incluye el momento, el método y la disposición de la solicitud HTTP. La instrumentación adicional crea subsegmentos en este segmento.

> **Note**
>
> En el AWS Lambda caso de las funciones, Lambda crea un segmento para cada solicitud muestreada. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

El siguiente ejemplo intercepta solicitudes en el puerto 8000 y devuelve “Hello!” como respuesta. Crea el segmento `myApp` e instrumenta llamadas a través de cualquier aplicación.

Example main.go

```
func main() {
  http.Handle("/", xray.Handler(xray.NewFixedSegmentNamer("MyApp"),
 http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello!"))
  })))

  http.ListenAndServe(":8000", nil)
}
```

Cada segmento tiene un nombre que identifica la aplicación en el mapa de servicio. El nombre del segmento se puede asignar de forma estática o se puede configurar el SDK para que le asigne un nombre dinámico en función del encabezado del host de la solicitud entrante. La nomenclatura dinámica permite agrupar los rastros en función del nombre de dominio de la solicitud y aplicar un nombre predeterminado si el nombre no coincide con el patrón esperado (por ejemplo, si el encabezado del host está falsificado).

> **Solicitudes reenviadas**
>
> Si un equilibrador de carga u otro intermediario reenvía una solicitud a la aplicación, X-Ray toma la IP de cliente del encabezado `X-Forwarded-For` de la solicitud en lugar de tomar la IP de origen del paquete IP. La IP de cliente que se graba para una solicitud reenviada puede estar falsificada, por lo que no se debe confiar en ella.

Cuando se reenvía una solicitud, el SDK crea un campo adicional en el segmento para indicar que se realizó esta acción. Si el segmento contiene el campo `x_forwarded_for` configurado en `true`, el IP del cliente se obtiene a partir del encabezado `X-Forwarded-For` en la solicitud HTTP.

El controlador crea un segmento para cada solicitud entrante con un bloque `http` que contiene la siguiente información:

- Método HTTP: GET, POST, PUT, DELETE, etc.
- Dirección del cliente: la dirección IP del cliente que envió la solicitud.
- Código de respuesta: el código de respuesta HTTP para la solicitud finalizada.
- Intervalo: la hora de inicio (cuando se recibió la solicitud) y la hora de finalización (cuando se envió la respuesta).

- Agente del usuario: el `user-agent` de la solicitud.
- Longitud del contenido: la `content-length` de la respuesta.

Configuración de una estrategia de nomenclatura de segmentos

AWS X-Ray utiliza un nombre de servicio para identificar la aplicación y distinguirla del resto de aplicaciones, bases de datos, recursos externos APIs y otros AWS recursos que utiliza la aplicación. Cuando el SDK de X-Ray genera segmentos para las solicitudes entrantes, registra el nombre del servicio de la aplicación en el campo de nombre del segmento.

El SDK de X-Ray puede nombrar los segmentos utilizando el nombre de host en el encabezado de la solicitud HTTP. Sin embargo, este encabezado se puede falsificar, lo que podría provocar nodos inesperados en el mapa de servicio. Para evitar que el SDK nombre los segmentos de forma incorrecta debido a que las solicitudes tienen encabezados de host falsificados, debe especificar un nombre predeterminado para las solicitudes entrantes.

Si la aplicación atiende solicitudes de varios dominios, puede configurar el SDK para que utilice una estrategia de nomenclatura dinámica que refleje esto en los nombres de los segmentos. Una estrategia de nomenclatura dinámica permite al SDK usar el nombre de host para las solicitudes que coinciden con un patrón esperado y aplicar el nombre predeterminado a las solicitudes que no coincidan.

Por ejemplo, es posible que tenga una sola aplicación que atienda solicitudes a tres subdominios: `www.example.com`, `api.example.com` y `static.example.com`. Puede usar una estrategia de nomenclatura dinámica con el patrón `*.example.com` para identificar los segmentos de cada subdominio con un nombre diferente, lo que da como resultado tres nodos de servicio en el mapa de servicio. Si su aplicación recibe solicitudes con un nombre de host que no coincide con el patrón, verá un cuarto nodo en el mapa de servicio con el nombre alternativo que especifique.

Si desea utilizar el mismo nombre para todos los segmentos de solicitud, especifique el nombre de la aplicación cuando cree el controlador, tal y como se muestra en la sección anterior.

> **Note**
>
> Puede anular el nombre de servicio predeterminado que ha definido en el código mediante la `AWS_XRAY_TRACING_NAME`variable de entorno???.

Una estrategia de nomenclatura dinámica define un patrón con el que deben coincidir los nombres de host y un nombre predeterminado que se utiliza si el nombre de host de la solicitud HTTP no coincide con el patrón. Para nombrar segmentos de forma dinámica, utilice `NewDynamicSegmentNamer` para configurar el nombre predeterminado y el patrón que coincida.

Example main.go

Si el nombre de host de la solicitud coincide con el patrón `*.example.com`, utilice el nombre de host. De lo contrario, utilice `MyApp`.

```
func main() {
  http.Handle("/", xray.Handler(xray.NewDynamicSegmentNamer("MyApp", "*.example.com"),
 http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello!"))
  })))

  http.ListenAndServe(":8000", nil)
}
```

Rastreo de llamadas AWS del SDK con el X-Ray SDK for Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando tu aplicación realiza llamadas Servicios de AWS para almacenar datos, escribir en una cola o enviar notificaciones, el X-Ray SDK for Go rastrea las llamadas en sentido descendente en subsegmentos. El rastreo Servicios de AWS y los recursos a los que accede dentro de esos servicios (por ejemplo, un bucket de Amazon S3 o una cola de Amazon SQS) aparecen como nodos descendentes en el mapa de rastreo de la consola X-Ray.

Para rastrear clientes del SDK de AWS, envuelva el objeto de cliente con la llamada `xray.AWS()`, tal y como se muestra en el siguiente ejemplo.

Example main.go

```
var dynamo *dynamodb.DynamoDB
func main() {
  dynamo = dynamodb.New(session.Must(session.NewSession()))
  xray.AWS(dynamo.Client)
}
```

A continuación, cuando utilice el cliente del SDK de AWS, utilice la versión `withContext` del método de llamada y transfiera el `context` desde el objeto `http.Request` transferido al handler.

Example main.go: llamada al SDK AWS

```
func listTablesWithContext(ctx context.Context) {
  output := dynamo.ListTablesWithContext(ctx, &dynamodb.ListTablesInput{})
  doSomething(output)
}
```

Para todos los servicios, puede ver el nombre de la API a la que se llama en la consola de X-Ray. Para un subconjunto de servicios, el SDK de X-Ray agrega información al segmento para proporcionar una mayor granularidad en el mapa de servicio.

Por ejemplo, cuando realiza una llamada con un cliente instrumentado de DynamoDB, el SDK agrega el nombre de tabla al segmento para las llamadas que se dirigen a una tabla. En la consola, cada tabla aparece como nodo independiente en el mapa de servicio, con un nodo genérico de DynamoDB para las llamadas que no se dirigen a una tabla.

Example Subsegmento para una llamada a DynamoDB con el fin de guardar un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
```

```
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Cuando accede a recursos designados, las llamadas a los siguientes servicios crean nodos adicionales en el mapa de servicio. Las llamadas que no están dirigidas a recursos concretos crean un nodo genérico en el servicio.

- Amazon DynamoDB: nombre de tabla
- Amazon Simple Storage Service: nombre de bucket y de clave
- Amazon Simple Queue Service: nombre de cola

Rastreo de llamadas a servicios web HTTP posteriores con el SDK de X-Ray para Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando tu aplicación realiza llamadas a microservicios o a HTTP públicos APIs, puedes utilizarlos *`xray.Client`* para instrumentar esas llamadas como subsegmentos de tu aplicación Go, como se muestra en el siguiente ejemplo, donde http-client es un cliente HTTP.

El cliente crea una copia superficial del cliente HTTP proporcionado, tomando como valor predeterminado `http.DefaultClient`, con roundtripper envuelto con `xray.RoundTripper`.

Example

<caption>main.go: cliente HTTP</caption>

```
myClient := xray.Client(http-client)
```

<caption>main.go: rastro de una llamada HTTP posterior con la biblioteca ctxhttp</caption>

En el siguiente ejemplo se instrumenta la llamada HTTP saliente con la biblioteca ctxhttp mediante `xray.Client`. `ctx` se puede transferir desde la llamada precedente. Eso garantiza que se utilice el contexto de segmento existente. Por ejemplo, X-Ray no permite crear un nuevo segmento dentro de una función de Lambda, por lo que se debe utilizar el contexto de segmento de Lambda existente.

```
resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

Rastreo de consultas SQL con el SDK de X-Ray para Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Para rastrear llamadas de SQL a PostgreSQL o MySQL, sustituyendo llamadas de `sql.Open` a `xray.SQLContext`, tal y como se muestra en el siguiente ejemplo. Si es posible, URLs utilícelas en lugar de cadenas de configuración.

Example main.go

```
func main() {
  db, err := xray.SQLContext("postgres", "postgres://user:password@host:port/db")
  row, err := db.QueryRowContext(ctx, "SELECT 1") // Use as normal
}
```

Generación de subsegmentos personalizados con el SDK de X-Ray para Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

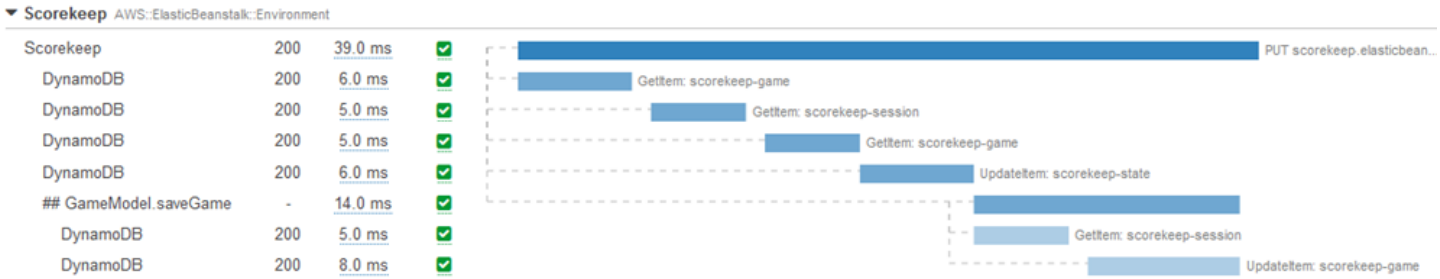
Los subsegmentos amplían el segmento de un rastro con detalles sobre el trabajo realizado para atender una solicitud. Cada vez que usted realiza una llamada con un cliente instrumentado, el SDK de X-Ray registra la información generada en un subsegmento. Puede crear subsegmentos adicionales para agrupar otros subsegmentos, medir el rendimiento de una sección de código o registrar anotaciones y metadatos.

Utilice el método `Capture` para crear un subsegmento en torno a una función.

Example main.go: subsegmento personalizado

```
func criticalSection(ctx context.Context) {
  //this is an example of a subsegment
  xray.Capture(ctx, "GameModel.saveGame", func(ctx1 context.Context) error {
    var err error

    section.Lock()
    result := someLockedResource.Go()
    section.Unlock()

    xray.AddMetadata(ctx1, "ResourceResult", result)
  })
```

En la siguiente captura de pantalla se muestra un ejemplo de cómo podría aparecer el subsegmento `saveGame` en rastreos para la aplicación `Scorekeep`.



Adición de anotaciones y metadatos a los segmentos con el SDK de X-Ray para Go

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede registrar información adicional acerca de las solicitudes, el entorno o su aplicación con anotaciones y metadatos. Puede añadir anotaciones y metadatos a los segmentos que crea el SDK de X-Ray o a los subsegmentos personalizados que cree usted mismo.

Las anotaciones son pares de clave-valor con valores de cadena, numéricos o booleanos. Las anotaciones se indexan para su uso con expresiones de filtro. Utilice anotaciones para registrar los datos que desee utilizar para agrupar rastros en la consola o cuando llame a la API de `GetTraceSummaries`.

Los metadatos son pares de clave-valor con valores de cualquier tipo, por ejemplo objetos y listas, pero que no se indexan para utilizarlos con expresiones de filtro. Utilice los metadatos para registrar datos adicionales que desee almacenar en el rastro, pero que no necesite usar para hacer búsquedas.

Además de anotaciones y metadatos, también puede registrar cadenas de ID de usuario en los segmentos. IDs Los usuarios se registran en un campo independiente en los segmentos y se indexan para usarlos en la búsqueda.

Secciones

Registro de anotaciones con el SDK de X-Ray para Go

Utilice anotaciones para registrar información sobre segmentos que desee indexar para las búsquedas.

Requisitos de anotación

- Claves: la clave de una anotación de X-Ray puede contener hasta 500 caracteres alfanuméricos. No se pueden usar espacios ni símbolos, excepto el punto (.)
- Valores: el valor de una anotación de X-Ray puede contener hasta 1000 caracteres Unicode.
- Número de anotaciones: se pueden utilizar hasta 50 anotaciones por rastro.

Para grabar anotaciones, llame a `AddAnnotation` con una cadena que contenga los metadatos que desea asociar con el segmento.

```
xray.AddAnnotation(key string, value interface{})
```

El SDK registra las anotaciones como pares de clave-valor en un objeto `annotations` del documento de segmento. Si llama dos veces a `AddAnnotation` con la misma clave, se sobrescriben los valores previamente registrados en el mismo segmento.

Para encontrar rastros que tengan anotaciones con valores específicos, utilice la palabra clave `annotation[key]` en una expresión de filtro.

Registro de metadatos con el SDK de X-Ray para Go

Utilice los metadatos para registrar información sobre segmentos que no necesite indexar para las búsquedas.

Para registrar metadatos, llame a `AddMetadata` con una cadena que contenga los metadatos que desea asociar con el segmento.

```
xray.AddMetadata(key string, value interface{})
```

Grabación del usuario IDs con el X-Ray SDK for Go

Registre IDs el usuario en los segmentos de solicitud para identificar al usuario que envió la solicitud.

Para registrar al usuario IDs

1. Obtenga una referencia al segmento actual desde `AWSXRay`.

   ```
   import (
     "context"
     "github.com/aws/aws-xray-sdk-go/xray"
   )

   mySegment := xray.GetSegment(context)
   ```

2. Llame a `setUser` con un ID de cadena del usuario que envió la solicitud.

   ```
   mySegment.User = "U12345"
   ```

Para buscar rastros de un ID de usuario, utilice la palabra clave `user` en una expresión de filtro.

Uso de Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede instrumentar su aplicación Java de dos maneras para enviar rastros a X-Ray:

- AWS Distro for OpenTelemetry Java: una AWS distribución que proporciona un conjunto de bibliotecas de código abierto para enviar métricas y rastreos correlacionados a varias soluciones de AWS monitoreo, incluidas Amazon y Amazon OpenSearch Service CloudWatch AWS X-Ray, a través de AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK para Java: conjunto de bibliotecas para generar y enviar trazas a X-Ray mediante el daemon X-Ray.

Para obtener más información, consulte Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs.

AWS Distro para OpenTelemetry Java

Con AWS Distro para OpenTelemetry (ADOT) Java, puede instrumentar sus aplicaciones una vez y enviar métricas y rastros correlacionados a varias soluciones de supervisión de AWS, incluidas Amazon CloudWatch, AWS X-Ray y Amazon OpenSearch Service. El uso de X-Ray con ADOT requiere dos componentes: un SDK de OpenTelemetry habilitado para su uso con X-Ray y AWS Distro para OpenTelemetry Collector habilitado para su uso con X-Ray. ADOT Java admite la instrumentación automática, lo que permite a la aplicación del usuario enviar rastros sin cambios de código.

Para empezar, consulte la documentación de AWS Distro para OpenTelemetry Java.

Para obtener más información sobre el uso de AWS Distro para OpenTelemetry con AWS X-Ray y otros Servicios de AWS, consulte AWS Distro para OpenTelemetry o la documentación de AWS Distro para OpenTelemetry.

Para obtener información adicional sobre los lenguajes compatibles y el uso, consulte AWS Observability en GitHub.

AWS X-Ray SDK for Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Java es un conjunto de bibliotecas para aplicaciones web Java que proporcionan clases y métodos para generar y enviar datos de rastreo al daemon de X-Ray. Los datos de rastreo incluyen información sobre las solicitudes HTTP entrantes atendidas por la aplicación y las llamadas que la aplicación realiza a los servicios descendentes mediante el AWS SDK, los clientes HTTP o un conector de base de datos SQL. También puede crear segmentos de forma manual y agregar información de depuración en anotaciones y metadatos.

El SDK de X-Ray para Java es un proyecto de código abierto. Puedes seguir el proyecto y enviar las incidencias y solicitudes de cambios en GitHub: github. com/aws/aws- xray-sdk-java

Comience añadiendo `AWSXRayServletFilter` como filtro de servlet para rastrear las solicitudes entrantes. Los filtros de servlets crean un segmento. Mientras el segmento está abierto, puede utilizar los métodos del cliente del SDK para añadir información al segmento y crear subsegmentos para rastrear llamadas posteriores. El SDK también registra automáticamente las excepciones que produce su aplicación mientras el segmento está abierto.

A partir de la versión 1.3, puede instrumentar una aplicación mediante la programación orientada a aspectos (AOP) de Spring. Lo que esto significa es que puede instrumentar su aplicación, mientras está en ejecución AWS, sin añadir ningún código al tiempo de ejecución de la aplicación.

A continuación, utilice el SDK de X-Ray para Java para instrumentar a sus AWS SDK for Java clientes mediante la inclusión del submódulo SDK Instrumentor en la configuración de compilación. Cada vez que realizas una llamada a un recurso Servicio de AWS o a un servicio intermedio con un cliente instrumentado, el SDK registra la información sobre la llamada en un subsegmento. Servicios de AWS y los recursos a los que accedes desde los servicios aparecen como nodos descendentes en el mapa de rastreo para ayudarte a identificar los errores y los problemas de limitación en las conexiones individuales.

Si no quieres instrumentar todas las llamadas posteriores Servicios de AWS, puedes omitir el submódulo Instrumentor y elegir los clientes a los que quieres instrumentar. Instrumente a los clientes individuales añadiendo un `TracingHandler` a un cliente de servicio AWS del SDK.

Otros submódulos de X-Ray SDK for Java proporcionan instrumentación para llamadas posteriores a bases de datos HTTP web APIs y SQL. Puede utilizar las versiones de `HTTPClient` y `HTTPClientBuilder` del SDK de X-Ray para Java en el submódulo Apache HTTP para instrumentar clientes Apache HTTP. Para instrumentar consultas SQL, añada el interceptor del SDK al origen de datos.

En cuanto empiece a utilizar el SDK, personalice su comportamiento configurando la grabadora y el filtro de servlet. Puede añadir complementos para registrar los datos sobre los recursos informáticos que ejecutan su aplicación, personalizar el comportamiento de muestreo mediante la definición de reglas de muestreo y definir el nivel de log para ver más o menos información del SDK en los logs de las aplicaciones.

Registre información adicional acerca de las solicitudes y el trabajo que la aplicación realiza en anotaciones y metadatos. Las anotaciones son pares sencillos de clave-valor que se indexan para su uso con expresiones de filtro para poder buscar rastros que contengan datos específicos. Las entradas de metadatos son menos restrictivas y pueden registrar objetos y matrices completos, es decir, todo lo que se pueda serializar en JSON.

> **Anotaciones y metadatos**
>
> Las anotaciones y los metadatos son texto arbitrario que se agrega a los segmentos con el SDK de X-Ray. Las anotaciones se indexan para su uso con expresiones de filtro. Los metadatos no se indexan pero se pueden ver en el segmento sin procesar con la consola o la API de X-Ray. Cualquier persona a la que conceda acceso de lectura a X-Ray puede ver estos datos.

Si tiene un gran número de clientes instrumentados en su código, un único segmento de solicitud puede contener muchos subsegmentos, uno por cada llamada realizada con un cliente instrumentado. Puede organizar y agrupar los subsegmentos incluyendo las llamadas del cliente en subsegmentos personalizados. Puede crear un subsegmento personalizado para una función completa o para cualquier sección de código, y registrar los metadatos y las anotaciones en el subsegmento en lugar de escribirlo todo en el segmento principal.

Submódulos

Puede descargar el SDK de X-Ray para Java desde Maven. El SDK de X-Ray para Java está dividido en submódulos por caso de uso, con una factura de materiales para la administración de las versiones:

- `aws-xray-recorder-sdk-core` (obligatorio): funcionalidad básica para la creación y transmisión de segmentos. Incluye `AWSXRayServletFilter` para instrumentar solicitudes entrantes.
- `aws-xray-recorder-sdk-aws-sdk`— Instrumenta las llamadas que Servicios de AWS se realizan con AWS SDK for Java los clientes añadiendo un cliente de rastreo como gestor de solicitudes.
- `aws-xray-recorder-sdk-aws-sdk-v2`— Las llamadas de Instruments Servicios de AWS se realizan con clientes de la versión AWS SDK for Java 2.2 y versiones posteriores añadiendo un cliente de rastreo como interceptor de solicitudes.
- `aws-xray-recorder-sdk-aws-sdk-instrumentor`— Con, instrumenta a `aws-xray-recorder-sdk-aws-sdk` todos los clientes automáticamente. AWS SDK for Java
- `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor`— Con`aws-xray-recorder-sdk-aws-sdk-v2`, instrumenta todos los clientes de AWS SDK for Java 2.2 y versiones posteriores de forma automática.
- `aws-xray-recorder-sdk-apache-http`: instrumenta llamadas HTTP salientes realizadas con clientes Apache HTTP.
- `aws-xray-recorder-sdk-spring`: proporciona interceptores para las aplicaciones del marco de trabajo Spring AOP.
- `aws-xray-recorder-sdk-sql-postgres`: instrumenta llamadas salientes a la base de datos PostgreSQL realizadas con JDBC.
- `aws-xray-recorder-sdk-sql-mysql`: instrumenta llamadas salientes a la base de datos MySQL realizadas con JDBC.

- `aws-xray-recorder-sdk-bom`: proporciona una lista de materiales que puede utilizar para especificar la versión que se debe utilizar en todos los submódulos.
- `aws-xray-recorder-sdk-metrics`— Publica CloudWatch métricas de Amazon sin muestrear de tus segmentos de X-Ray recopilados.

Si utiliza Maven o Gradle para crear la aplicación, añada el SDK de X-Ray para Java a la configuración de compilación.

Para ver documentos de referencia sobre las clases y los métodos del SDK, consulte Referencia de la API de AWS X-Ray SDK para Java.

Requisitos

El X-Ray SDK for Java requiere Java 8 o posterior, Servlet API 3, el AWS SDK y Jackson.

El SDK depende de las siguientes bibliotecas durante la compilación y el tiempo de ejecución:

- AWS SDK para Java versión 1.11.398 o posterior
- Servlet API 3.1.0

Estas dependencias se declaran en el archivo `pom.xml` del SDK y se incluyen de forma automática si realiza la compilación con Maven o Gradle.

Si utiliza una biblioteca que se incluye en el SDK de X-Ray para Java debe utilizar la versión que se incluye. Por ejemplo, si ya depende de Jackson durante el tiempo de ejecución e incluye archivos JAR en la implementación de esa dependencia, debe eliminar dichos archivos porque el archivo JAR del SDK incluye sus propias versiones de las bibliotecas Jackson.

Administración de dependencias

El SDK de X-Ray para Java está disponible en Maven:

- Grupo: `com.amazonaws`
- Artefacto: `aws-xray-recorder-sdk-bom`
- Versión: `2.11.0`

Si utiliza Maven para compilar la aplicación, añada el SDK como dependencia en su archivo `pom.xml`.

Example pom.xml: dependencias

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-xray-recorder-sdk-bom</artifactId>
      <version>2.11.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-apache-http</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-aws-sdk</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-aws-sdk-instrumentor</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-sql-postgres</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-sql-mysql</artifactId>
  </dependency>
</dependencies>
```

Si utiliza Gradle, añada el SDK como dependencia de tiempo de compilación en su archivo `build.gradle`.

Example build.gradle: dependencias

```
dependencies {
  compile("org.springframework.boot:spring-boot-starter-web")
  testCompile("org.springframework.boot:spring-boot-starter-test")
  compile("com.amazonaws:aws-java-sdk-dynamodb")
  compile("com.amazonaws:aws-xray-recorder-sdk-core")
  compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk")
  compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk-instrumentor")
  compile("com.amazonaws:aws-xray-recorder-sdk-apache-http")
  compile("com.amazonaws:aws-xray-recorder-sdk-sql-postgres")
  compile("com.amazonaws:aws-xray-recorder-sdk-sql-mysql")
  testCompile("junit:junit:4.11")
}
dependencyManagement {
    imports {
        mavenBom('com.amazonaws:aws-java-sdk-bom:1.11.39')
        mavenBom('com.amazonaws:aws-xray-recorder-sdk-bom:2.11.0')
    }
}
```

Si utiliza Elastic Beanstalk para implementar la aplicación, puede utilizar Maven o Gradle para compilar la aplicación en la instancia cada vez que realice la implementación, en lugar de compilar y cargar un archivo de gran tamaño que incluya todas sus dependencias. Consulte la aplicación de ejemplo para ver un ejemplo en el que se utiliza Gradle.

Agente de instrumentación automática de AWS X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El agente de AWS X-Ray autoinstrumentación para Java es una solución de rastreo que instrumenta sus aplicaciones web Java con un mínimo esfuerzo de desarrollo. El agente permite rastrear las aplicaciones basadas en servlets y todas las solicitudes posteriores del agente realizadas con marcos y bibliotecas compatibles. Entre ellas se incluyen las solicitudes HTTP a Apache, solicitudes mediante el SDK de AWS y consultas en SQL posteriores realizadas con un controlador JDBC. El agente propaga el contexto de X-Ray, incluidos todos los segmentos y subsegmentos activos, a través de subprocesos. Todas las configuraciones y la versatilidad del SDK de X-Ray siguen disponibles con el agente para Java. Se eligieron los valores predeterminados adecuados para garantizar que el agente funcione con el mínimo esfuerzo.

Lo más adecuado para la solución del agente de X-Ray son los servidores de aplicaciones web Java de solicitud-respuesta basados en servlets. Si su aplicación utiliza un marco asíncrono o no está bien diseñada como servicio de solicitud-respuesta, la instrumentación manual con el SDK podría ser más conveniente.

El agente X-Ray se crea con el kit de herramientas de comprensión de sistemas distribuidos, o Di. SCo Di SCo es un marco de código abierto para crear agentes Java que se pueden usar en sistemas distribuidos. Si bien no es necesario entender a Di SCo para usar el agente de rayos X, puede obtener más información sobre el proyecto visitando su página de inicio en GitHub. El agente de X-Ray también es de código totalmente abierto. Para ver el código fuente, hacer contribuciones o plantear problemas sobre el agente, visita su repositorio en GitHub.

Aplicación de muestra

La aplicación eb-java-scorekeepde muestra está adaptada para ser instrumentada con el agente de rayos X. Esta ramificación no contiene ningún filtro de servlet ni configuración de grabadora, ya que estas funciones las realiza el agente. Para ejecutar la aplicación de forma local o utilizando recursos de AWS , siga los pasos indicados en el archivo readme de la aplicación de muestra. Las instrucciones para usar la aplicación de muestra con el fin de generar rastros de X-Ray se encuentran en el tutorial de la aplicación de muestra.

Introducción

Para empezar a utilizar el agente de instrumentación automática de X-Ray para Java en su propia aplicación, siga estos pasos.

1. Ejecute el daemon de X-Ray en su entorno. Para obtener más información, consulte AWS X-Ray demonio.

2. Descargue la última distribución del agente. Descomprima el archivo y tome nota de su ubicación en el sistema de archivos. El contenido debe ser similar al siguiente.

```
disco
### disco-java-agent.jar
### disco-plugins
    ### aws-xray-agent-plugin.jar
    ### disco-java-agent-aws-plugin.jar
    ### disco-java-agent-sql-plugin.jar
    ### disco-java-agent-web-plugin.jar
```

3. Modifique los argumentos de JVM de su aplicación para incluir lo siguiente y así habilitar el agente. Asegúrese de que el argumento `-javaagent` esté colocado delante del argumento `-jar`, si procede. El proceso de modificación de los argumentos de JVM varía en función de las herramientas y los marcos que utilice para lanzar su servidor de Java. Consulte la documentación del marco de su servidor para obtener orientación específica.

```
-javaagent:/<path-to-disco>/disco-java-agent.jar=pluginPath=/<path-to-disco>/disco-plugins
```

4. Para especificar cómo aparece el nombre de la aplicación en la consola de X-Ray, establezca la variable de entorno `AWS_XRAY_TRACING_NAME` o la propiedad de sistema `com.amazonaws.xray.strategy.tracingName`. Si no se proporciona un nombre, se utilizará un nombre predeterminado.
5. Reinicie el servidor o el contenedor. A partir de ese momento se rastrearán las solicitudes entrantes y sus llamadas posteriores. Si no ve los resultados esperados, consulte the section called “Solución de problemas”.

Configuración

El agente de X-Ray se configura mediante un archivo JSON externo proporcionado por el usuario. De forma predeterminada, este archivo se encuentra en la raíz de la ruta de clases del usuario denominada `xray-agent.json` (por ejemplo, en su directorio `resources`). Puede configurar una ubicación personalizada para el archivo de configuración estableciendo la propiedad de sistema `com.amazonaws.xray.configFile` en la ruta absoluta del sistema de archivos del archivo de configuración.

A continuación, se muestra un ejemplo de archivo de configuración.

```
{
    "serviceName": "XRayInstrumentedService",
    "contextMissingStrategy": "LOG_ERROR",
    "daemonAddress": "127.0.0.1:2000",
    "tracingEnabled": true,
    "samplingStrategy": "CENTRAL",
    "traceIdInjectionPrefix": "prefix",
    "samplingRulesManifest": "/path/to/manifest",
    "awsServiceHandlerManifest": "/path/to/manifest",
    "awsSdkVersion": 2,
    "maxStackTraceLength": 50,
    "streamingThreshold": 100,
    "traceIdInjection": true,
    "pluginsEnabled": true,
    "collectSqlQueries": false
}
```

Especificación de la configuración

En la tabla siguiente se describen valores válidos para cada propiedad. Los nombres de las propiedades hay distinción entre mayúsculas y minúsculas, pero en sus claves no. En el caso de propiedades que las variables de entorno y las propiedades del sistema pueden anular, el orden de prioridad es siempre el siguiente: primero la variable de entorno, luego la propiedad del sistema y, por último, el archivo de configuración. Para obtener información sobre las propiedades que puede anular, consulte Variables de entorno. Todos los campos son opcionales.

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
serviceName	Cadena	Cualquier cadena	El nombre del servicio instrumentado tal como aparecerá en la consola de X-Ray.	AWS_XRAY_TRACING_NAME	com.amazonaws.xray.strategy.tracingName	XRayInstrumentedService

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
contextMissingStrategy	Cadena	LOG_ERROR, IGNORE_ERROR	La acción que realiza el agente cuando intenta utilizar el contexto del segmento de X-Ray, pero no hay ninguno.	AWS_XRAY_CONTEXT_MISSING	com.amazonaws.xray.strategy.contextMissingStrategy	LOG_ERROR
daemonAddress	Cadena	Dirección IP y puerto formateados o lista de direcciones TCP y UDP	La dirección que utiliza el agente para comunicarse con el daemon de X-Ray.	AWS_XRAY_DAEMON_ADDRESS	com.amazonaws.xray.emitter.daemonAddress	127.0.0.1:2000
tracingEnabled	Booleano	True, False	Permite la instrumentación mediante el agente de X-Ray.	AWS_XRAY_TRACING_HABILITADO	com.amazonaws.xray.tracingEnabled	TRUE

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
samplingStrategy	Cadena	CENTRAL, LOCAL, NONE o ALL	La estrategia de muestreo que utiliza el agente. ALL captura todas las solicitudes, NONE no captura ninguna. Consulte Reglas de muestreo.	N/A	N/A	CENTRAL
traceIdInjectionPrefijo	Cadena	Cualquier cadena	Incluye el prefijo proporcionado antes de inyectar la traza IDs en los registros.	N/A	N/A	Ninguno (cadena vacía)

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
samplingRulesManifest	Cadena	Ruta de archivo absoluta	La ruta a un archivo de reglas de muestreo personalizadas que se utilizará como fuente de las reglas de muestreo para la estrategia de muestreo local o las reglas alternativas para la estrategia central.	N/A	N/A	DefaultSamplingRules.json

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
awsServiceHandlerManifiesto	Cadena	Ruta de archivo absoluta	La ruta a una lista de parámetros permitidos personalizados que captura información adicional de los clientes del SDK de AWS .	N/A	N/A	DefaultOperationParameterWhitelist.json
awsSdkVersion	Entero	1, 2	Versión del SDK de AWS para Java que está utilizando. Se ignora si no se establece también `awsServiceHandlerManifest` .	N/A	N/A	2

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
maxStackTraceLongitud	Entero	Enteros no negativos	El número máximo de líneas de un rastro de pila que se pueden registrar en una traza.	N/A	N/A	50
streamingThreshold	Entero	Enteros no negativos	Una vez cerrados al menos este número de subsegmentos, se transmiten al daemon out-of-band para evitar que los fragmentos sean demasiado grandes.	N/A	N/A	100

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
traceIdInjection	Booleano	True, False	Permite la inyección de los ID de rastro de X-Ray en los registros si también se agregan las dependencias y la configuración descritas en la configuración de registro. De lo contrario, no hace nada.	N/A	N/A	TRUE

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
pluginsEnabled	Booleano	True, False	Habilita los complementos que registran los metadatos sobre los AWS entornos en los que se opera. Consulte Complementos de servicio.	N/A	N/A	TRUE
collectSqlQueries	Booleano	True, False	Registra cadenas de consultas SQL en subsegmentos SQL en la medida de lo posible.	N/A	N/A	FALSO

Nombre de la propiedad	Tipo	Valores válidos	Description (Descripción)	Variable de entorno	Propiedad del sistema	Predeterminado
contextPropagation	Booleano	True, False	Propaga automáticamente el contexto de X-Ray entre subprocesos si es verdadero. De lo contrario, utiliza ThreadLocal para almacenar el contexto y se requiere la propagación manual de un subproceso a otro.	N/A	N/A	TRUE

Configuración de registro

El nivel de registro del agente de X-Ray se puede configurar de la misma manera que el SDK de X-Ray para Java. Consulte Registro para obtener más información sobre la configuración del registro con el SDK de X-Ray para Java.

Instrumentación manual

Si desea realizar la instrumentación manual además de la instrumentación automática del agente, añada el SDK de X-Ray como una dependencia a su proyecto. Tenga en cuenta que los filtros de servlets personalizados del SDK que se mencionan en Rastreo de solicitudes entrantes no son compatibles con el agente de X-Ray.

> **Note**
>
> Debe usar la última versión del SDK de X-Ray para realizar la instrumentación manual y usar el agente al mismo tiempo.

Si está trabajando en un proyecto de Maven, añada las siguientes dependencias a su archivo `pom.xml`.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
    <version>2.11.0</version>
  </dependency>
  </dependencies>
```

Si está trabajando en un proyecto de Gradle, añada las siguientes dependencias a su archivo `build.gradle`.

```
implementation 'com.amazonaws:aws-xray-recorder-sdk-core:2.11.0'
```

Puede añadir subsegmentos personalizados además de anotaciones, metadatos y usuarios IDs mientras usa el agente, tal como lo haría con el SDK normal. El agente propaga automáticamente el contexto de un subproceso a otro, por lo que no deberían hacer falta soluciones provisionales para propagar el contexto cuando se trabaja con aplicaciones multiproceso.

Solución de problemas

Como el agente ofrece una instrumentación totalmente automática, puede resultar difícil identificar la causa raíz de un problema cuando se están produciendo problemas. Si el agente de X-Ray no funciona según lo esperado, revise los siguientes problemas y soluciones. El agente y el SDK de X-Ray utilizan Jakarta Commons Logging (JCL). Para ver el resultado del registro, asegúrese de

que un puente que conecte JCL con su backend de registro esté en la ruta de clases, como en el siguiente ejemplo: `log4j-jcl` o `jcl-over-slf4j`.

Problema: he habilitado el agente para Java en mi aplicación, pero no veo nada en la consola de X-Ray

¿El daemon de X-Ray se está ejecutando en la misma máquina?

Si no es así, consulte la documentación del daemon de X-Ray para configurarlo.

¿Ve un mensaje similar a “Iniciando la grabadora del agente de X-Ray” en los registros de su aplicación?

Si ha agregado correctamente el agente a la aplicación, este mensaje estará registrado en el nivel INFO cuando se inicie la aplicación, antes de que esta comience a recibir solicitudes. Si este mensaje no aparece, significa que el agente para Java no se está ejecutando con el proceso de Java. Asegúrese de haber seguido todos los pasos de configuración correctamente sin errores tipográficos.

En los registros de tu aplicación, ¿ves varios mensajes de error que digan algo así como «Suprimir la excepción por falta de AWS X-Ray contexto»?

Estos errores se producen porque el agente está intentando instrumentar las solicitudes posteriores, como las solicitudes AWS del SDK o las consultas de SQL, pero no ha podido crear un segmento automáticamente. Si ve muchos de estos errores, es posible que el agente no sea la mejor herramienta para el uso que usted le quiere dar y, en su lugar, tal vez debería considerar la instrumentación manual con el SDK de X-Ray. Como alternativa, puede habilitar los registros de depuración del SDK de X-Ray para ver el rastro de pila del punto donde se producen las excepciones de falta de contexto. Puede agrupar estas porciones del código con segmentos personalizados, lo que debería corregir estos errores. Para ver un ejemplo de cómo encapsular las solicitudes posteriores con segmentos personalizados, consulte el código de muestra que aparece en Instrumentación de código de inicio.

Problema: algunos de los segmentos que espero no aparecen en la consola de X-Ray

¿Su aplicación utiliza el multiproceso?

Si espera que se creen ciertos segmentos pero no aparecen en la consola, la causa pueden ser subprocesos en segundo plano en la aplicación. Si su aplicación realiza tareas mediante subprocesos en segundo plano que son «activarlos y olvidarlos», como realizar una llamada puntual a una función de Lambda con AWS el SDK o sondear periódicamente algún punto final HTTP, esto

puede confundir al agente mientras propaga el contexto entre los subprocesos. Para comprobar que este es su problema, habilite los registros de depuración del SDK de X-Ray y mire a ver si hay mensajes como este: Not emitting segment named <NOMBRE > as it parents in-progress subsegments. Para solucionar este problema, puede intentar unir los subprocesos en segundo plano antes de que el servidor vuelva a funcionar para asegurarse de que todo el trabajo realizado en ellos queda registrado. O bien, puede establecer la configuración `contextPropagation` del agente en `false` para inhabilitar la propagación del contexto en subprocesos en segundo plano. En ese caso, tendrá que instrumentar manualmente esos subprocesos con segmentos personalizados o ignorar las excepciones de falta de contexto que generen.

¿Ha establecido reglas de muestreo?

Si aparecen segmentos aparentemente aleatorios o inesperados en la consola de X-Ray, o los segmentos que esperaba que estuvieran en la consola no están, es posible que tenga un problema de muestreo. El agente de X-Ray lleva a cabo un muestreo centralizado en todos los segmentos que crea, aplicando las reglas de la consola de X-Ray. La regla por defecto es obtener muestras en 1 segmento por segundo y, posteriormente, en el 5 % de los segmentos. Eso significa que es posible que no se obtengan muestras de los segmentos que se creen rápidamente con el agente. Para solucionar este problema, debe crear reglas de muestreo personalizadas en la consola de X-Ray para un muestreo adecuado de los segmentos deseados. Para obtener más información, consulte Configuración de reglas de muestreo.

Configuración del SDK de X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Java incluye una clase denominada `AWSXRay` que proporciona la grabadora global. Esto es un `TracingHandler` que puede utilizar para instrumentar su código. Puede

configurar la grabadora global para que personalice el `AWSXRayServletFilter` que crea los segmentos para las llamadas HTTP entrantes.

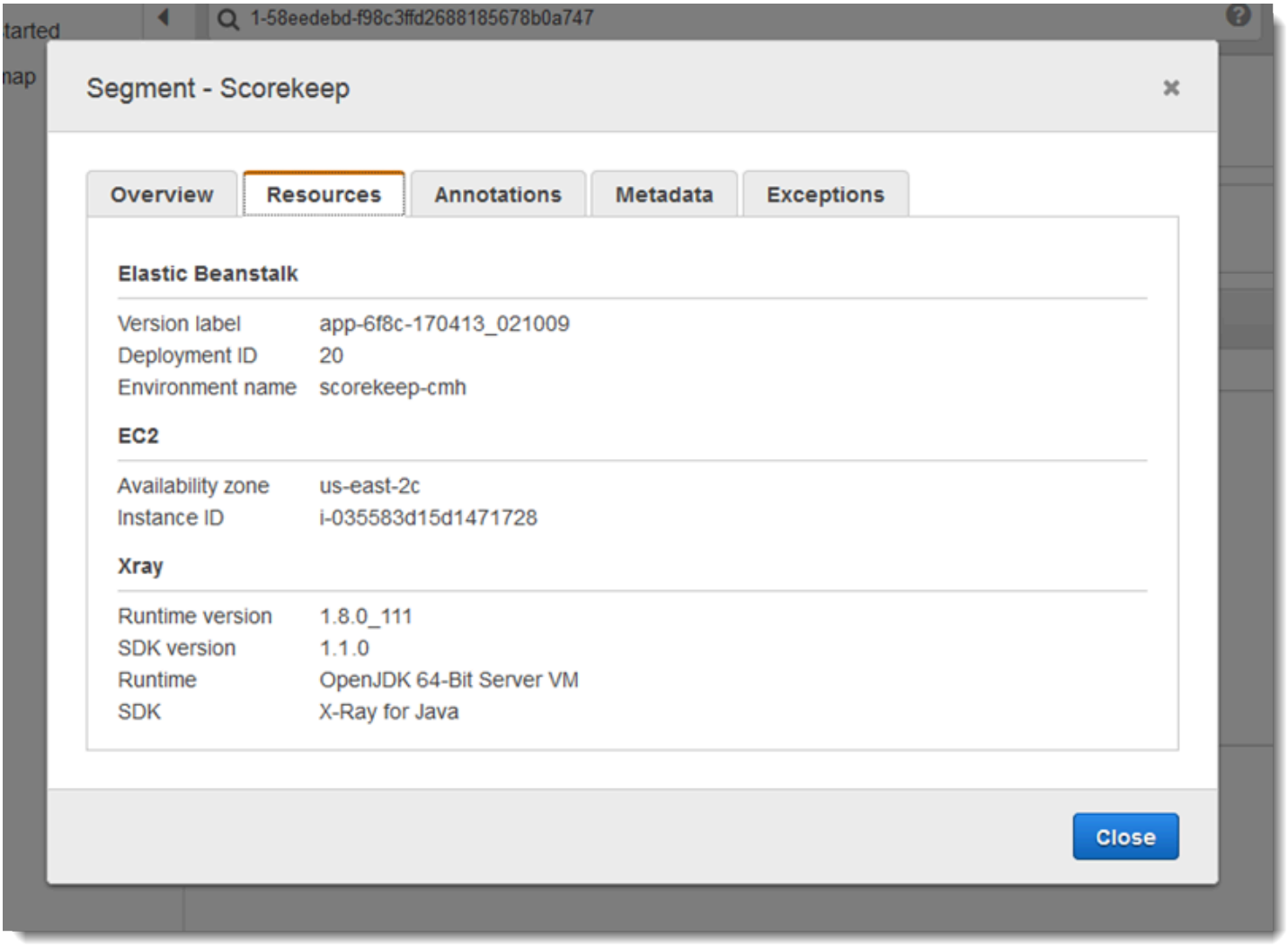
Secciones

Complementos del servicio

Utilice `plugins` para registrar información sobre el servicio que aloja la aplicación.

Plugins

- Amazon EC2 : `EC2Plugin` agrega el ID de la instancia, la zona de disponibilidad y el grupo de CloudWatch registros.
- Elastic Beanstalk: `ElasticBeanstalkPlugin` añade el nombre de entorno, la etiqueta de versión y el ID de implementación.
- Amazon ECS: `ECSPlugin` agrega el ID de contenedor.
- Amazon EKS: `EKSPlugin` añade el ID del contenedor, el nombre del clúster, el ID del pod y el grupo de CloudWatch registros.



Para utilizar un complemento, llame a `withPlugin` en su `AWSXRayRecorderBuilder`.

Example src/main/java/scorekeep/WebConfig.java: grabadora

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.plugins.ElasticBeanstalkPlugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

@Configuration
public class WebConfig {
...
  static {
    AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
 EC2Plugin()).withPlugin(new ElasticBeanstalkPlugin());
```

```
    URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
    builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

    AWSXRay.setGlobalRecorder(builder.build());
  }
}
```

El SDK también usa la configuración del complemento para establecer el campo `origin` en el segmento. Indica el tipo de AWS recurso que ejecuta la aplicación. Cuando utilizas varios complementos, el SDK utiliza el siguiente orden de resolución para determinar el origen: ElasticBeanstalk > EKS > ECS > EC2.

Reglas de muestreo

El SDK utiliza las reglas de muestreo que define el usuario en la consola de X-Ray para determinar qué solicitudes registrar. La regla predeterminada rastrea la primera solicitud cada segundo y el 5 % de las solicitudes adicionales de todos los servicios que envían rastros a X-Ray. Cree reglas adicionales en la consola de X-Ray para personalizar la cantidad de datos registrados para cada una de sus aplicaciones.

El SDK aplica las reglas personalizadas en el orden en que se definen. Si una solicitud coincide con varias reglas personalizadas, el SDK solo aplica la primera regla.

> **Note**
>
> Si el SDK no puede comunicarse con X-Ray para obtener las reglas de muestreo, vuelve a una regla local predeterminada de la primera solicitud cada segundo y del 5 % de las solicitudes adicionales por host. Esto puede ocurrir si el host no tiene permiso para llamar al muestreo APIs o no puede conectarse al daemon X-Ray, que actúa como proxy TCP para las llamadas a la API realizadas por el SDK.

También puede configurar el SDK para que cargue las reglas de muestreo desde un documento JSON. El SDK puede usar las reglas locales como respaldo para los casos en que el muestreo de X-Ray no esté disponible, o puede usar las reglas locales exclusivamente.

Example sampling-rules.json

```
{
```

```
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

En este ejemplo se define una regla personalizada y una regla predeterminada. La regla personalizada aplica un porcentaje de muestreo del 5 % sin un número mínimo de solicitudes de rastreo para las rutas incluidas bajo `/api/move/`. La regla predeterminada rastrea la primera solicitud cada segundo y el 10 % de las solicitudes adicionales.

La desventaja de definir las reglas localmente es que el objetivo establecido lo aplica cada instancia de la grabadora de forma independiente, en lugar de ser administrado por el servicio de X-Ray. A medida que se implementan más hosts, el porcentaje establecido se multiplica, lo que dificulta el control de la cantidad de datos registrados.

Sí AWS Lambda, no puedes modificar la frecuencia de muestreo. Si un servicio instrumentado llama a su función, Lambda registrará las llamadas que generaron solicitudes muestreadas por ese servicio. Si el rastreo activo está activado y no hay ningún encabezado de rastreo, Lambda toma la decisión de muestreo.

Para proporcionar reglas de copia de seguridad en Spring, configure la grabadora global con una `CentralizedSamplingStrategy` en una clase de configuración:

Example src/main/java/myapp/WebConfig.java: configuración de la grabadora

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;
```

```
@Configuration
public class WebConfig {

  static {
  AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
 EC2Plugin());

  URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
  builder.withSamplingStrategy(new CentralizedSamplingStrategy(ruleFile));

  AWSXRay.setGlobalRecorder(builder.build());
}
```

Para Tomcat, añada un agente de escucha que amplíe `ServletContextListener` y registre el agente de escucha en el descriptor de la implementación.

Example src/com/myapp/web/Startup.java

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

import java.net.URL;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class Startup implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent event) {
        AWSXRayRecorderBuilder builder =
 AWSXRayRecorderBuilder.standard().withPlugin(new EC2Plugin());

        URL ruleFile = Startup.class.getResource("/sampling-rules.json");
        builder.withSamplingStrategy(new CentralizedSamplingStrategy(ruleFile));

        AWSXRay.setGlobalRecorder(builder.build());
    }

    @Override
    public void contextDestroyed(ServletContextEvent event) { }
```

```
}
```

Example WEB-INF/web.xml

```
...
  <listener>
    <listener-class>com.myapp.web.Startup</listener-class>
  </listener>
```

Para utilizar reglas locales solo, reemplace la `CentralizedSamplingStrategy` por `LocalizedSamplingStrategy`.

```
builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));
```

Registro

De forma predeterminada, el SDK envía mensajes de nivel de `ERROR` a los registros de la aplicación. Puede habilitar el registro a nivel de depuración en el SDK para enviar registros más detallados al archivo de registro de la aplicación. Los niveles de registro válidos son `DEBUG`, `INFO`, `WARN`, `ERROR` y `FATAL`. El nivel de registro `FATAL` silencia todos los mensajes de registro porque el SDK no registra un nivel fatal.

Example application.properties

Configure el nivel de registro con la propiedad `logging.level.com.amazonaws.xray`.

```
logging.level.com.amazonaws.xray = DEBUG
```

Utilice registros de depuración para identificar problemas como la presencia de subsegmentos sin cerrar al generar subsegmentos de forma manual.

Inyección de ID de registro de seguimiento en los registros

Para exponer el ID de seguimiento completo actual en las instrucciones del registro, puede incluir el ID en el contexto de diagnóstico asignado (MDC). Mediante la interfaz `SegmentListener`, se llama a los métodos desde la grabadora de X-Ray durante los eventos del ciclo de vida del segmento. Cuando comienza un segmento o subsegmento, el ID de seguimiento completo se incluye en el MDC con la clave `AWS-XRAY-TRACE-ID`. Cuando termina ese segmento, se elimina la clave del MDC. Esto expone el ID de registro de seguimiento a la biblioteca de registro en uso. Cuando termina un subsegmento, su ID principal se incluye en el MDC.

Example ID de seguimiento completo

El ID completo se representa como `TraceID@EntityID`

```
1-5df42873-011e96598b447dfca814c156@541b3365be3dafc3
```

Esta función funciona con aplicaciones Java equipadas con el SDK de AWS X-Ray para Java y admite las siguientes configuraciones de registro:

- SLF4API de interfaz J con backend Logback
- SLF4API de front-end J con backend Log4J2
- API front-end Log4J2 con backend Log4J2

Consulte las siguientes pestañas para conocer las necesidades de cada front end y backend.

SLF4J Frontend

1. Agregue la siguiente dependencia de Maven a su proyecto.

   ```
   <dependency>
       <groupId>com.amazonaws</groupId>
       <artifactId>aws-xray-recorder-sdk-slf4j</artifactId>
       <version>2.11.0</version>
   </dependency>
   ```

2. Incluya el método `withSegmentListener` al construir la `AWSXRayRecorder`. Esto agrega una `SegmentListener` clase, que inyecta automáticamente una nueva traza en el J MDC. IDs SLF4

   `SegmentListener` toma una cadena opcional como parámetro para configurar el prefijo de la instrucción de registro. El prefijo se puede configurar de las siguientes maneras:

   - Ninguno: utiliza el prefijo predeterminado `AWS-XRAY-TRACE-ID`.
   - Vacío: utiliza una cadena vacía (por ejemplo, `""`).
   - Personalizado: utiliza un prefijo personalizado tal como se define en la cadena.

   Example **`AWSXRayRecorderBuilder`** instrucción

   ```
   AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
   ```

```
        .standard().withSegmentListener(new SLF4JSegmentListener("CUSTOM-
PREFIX"));
```

Log4J2 front end

1. Agregue la siguiente dependencia de Maven a su proyecto.

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-log4j</artifactId>
    <version>2.11.0</version>
</dependency>
```

2. Incluya el método `withSegmentListener` al construir la `AWSXRayRecorder`. Esto añadirá una `SegmentListener` clase, que inyectará automáticamente una nueva traza totalmente cualificada IDs en el SLF4 J MDC.

   `SegmentListener` toma una cadena opcional como parámetro para configurar el prefijo de la instrucción de registro. El prefijo se puede configurar de las siguientes maneras:

   - Ninguno: utiliza el prefijo predeterminado `AWS-XRAY-TRACE-ID`.
   - Vacío: utiliza una cadena vacía (por ejemplo, `""`) y elimina el prefijo.
   - Personalizado: utiliza el prefijo personalizado definido en la cadena.

   Example **`AWSXRayRecorderBuilder`** instrucción

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new Log4JSegmentListener("CUSTOM-
PREFIX"));
```

Logback backend

Para insertar el ID de registro de seguimiento en los eventos de registro, debe modificar el `PatternLayout` del registrador, que da formato a cada instrucción de registro.

1. Busque dónde está configurado el `patternLayout`. Puede hacerlo mediante programación o a través de un archivo de configuración XML. Para obtener más información, consulte Configuración de Logback.

2. Inserte `%X{AWS-XRAY-TRACE-ID}` en cualquier lugar del `patternLayout` para insertar el ID de rastro en las futuras instrucciones de registro. `%X{}` indica que recupera un valor con la clave proporcionada del MDC. Para obtener más información sobre PatternLayouts Logback, consulte. PatternLayout

Log4J2 backend

1. Busque dónde está configurado el `patternLayout`. Puede hacerlo mediante programación o a través de un archivo de configuración escrito en XML, JSON, YAML o con las propiedades del formato.

   Para obtener más información sobre la configuración de Log4J2 ´mediante un archivo de configuración, consulte Configuración.

   Para obtener más información sobre la configuración de Log4J2 mediante programación, consulte Configuración programática.
2. Inserte `%X{AWS-XRAY-TRACE-ID}` en cualquier lugar del `PatternLayout` para insertar el ID de rastro en las futuras instrucciones de registro. `%X{}` indica que recupera un valor con la clave proporcionada del MDC. Para obtener más información sobre PatternLayouts Log4J2, consulte Diseño de patrones.

Ejemplo de inyección de ID de registro de seguimiento

A continuación se muestra una cadena de `PatternLayout` modificada para incluir el ID de registro de seguimiento. El ID de registro de seguimiento se imprime después del nombre del subproceso (`%t`) y antes del nivel de registro (`%-5p`).

Example **`PatternLayout`** Con inyección de ID

```
%d{HH:mm:ss.SSS} [%t] %X{AWS-XRAY-TRACE-ID} %-5p %m%n
```

AWS X-Ray imprime automáticamente la clave y el ID de rastreo en la declaración de registro para facilitar el análisis. A continuación se muestra una instrucción de registro que utiliza el `PatternLayout` modificado.

Example Instrucción de registro con inyección de ID

```
2019-09-10 18:58:30.844 [nio-5000-exec-4]  AWS-XRAY-TRACE-ID:
 1-5d77f256-19f12e4eaa02e3f76c78f46a@1ce7df03252d99e1 WARN 1 - Your logging message
 here
```

El mensaje de registro en sí está alojado en el patrón `%m` y se establece al llamar al registrador.

Agentes de escucha de segmento

Los agentes de escucha de segmento son una interfaz para interceptar eventos del ciclo de vida, como el comienzo y el final de segmentos producidos por `AWSXRayRecorder`. La implementación de una función de evento de un oyente de segmento es posible que sea agregar la misma anotación a todos los subsegmentos cuando se crean con `onBeginSubsegment`, registrar un mensaje después de que cada segmento se envíe al daemon mediante `afterEndSegment` o registrar consultas enviadas por los interceptores de SQL mediante `beforeEndSubsegment` para verificar si el subsegmento representa una consulta SQL, agregando metadatos adicionales si es así.

Para consultar la lista completa de funciones de `SegmentListener`, consulte la documentación del SDK de grabación de AWS X-Ray para la API de Java.

En el ejemplo siguiente se muestra cómo agregar una anotación coherente a todos los subsegmentos al crear con `onBeginSubsegment` e imprimir un mensaje de registro al final de cada segmento con `afterEndSegment`.

Example MySegmentListener.java

```
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
import com.amazonaws.xray.listeners.SegmentListener;

public class MySegmentListener implements SegmentListener {
    .....

    @Override
    public void onBeginSubsegment(Subsegment subsegment) {
        subsegment.putAnnotation("annotationKey", "annotationValue");
    }

    @Override
    public void afterEndSegment(Segment segment) {
```

```
        // Be mindful not to mutate the segment
        logger.info("Segment with ID " + segment.getId());
    }
}
```

A continuación, se hace referencia a este agente de escucha de segmento personalizado cuando se crea `AWSXRayRecorder`.

Example AWSXRayRecorderBuilder declaración

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new MySegmentListener());
```

Variables de entorno

Puede usar variables de entorno para configurar el SDK de X-Ray para Java. El SDK admite las siguientes variables.

- `AWS_XRAY_CONTEXT_MISSING`: establezca esta opción en `RUNTIME_ERROR` para generar excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.

  Valores válidos

  - `RUNTIME_ERROR`: lance una excepción de tiempo de ejecución.
  - `LOG_ERROR`: registre un error y continúe (predeterminado).
  - `IGNORE_ERROR`: ignore el error y continúe.

  Se pueden producir errores relativos a segmentos o subsegmentos inexistentes al intentar usar un cliente instrumentado en el código de inicio que se ejecuta cuando no hay ninguna solicitud abierta o en el código que inicia un nuevo subproceso.
- `AWS_XRAY_DAEMON_ADDRESS`: establezca el host y el puerto del oyente del daemon de X-Ray. De forma predeterminada, el SDK utiliza `127.0.0.1:2000` tanto para los datos de rastro (UDP) como para el muestreo (TCP). Use esta variable si ha configurado el daemon para que escuche en un puerto diferente o si se ejecuta en un host diferente.

  Formato

  - El mismo puerto: *`address`*`:`*`port`*
  - Puertos diferentes: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*

- `AWS_LOG_GROUP`: establece el nombre de un grupo de registro en el grupo de registro asociado a su aplicación. Si su grupo de registros usa la misma AWS cuenta y región que su aplicación, X-Ray buscará automáticamente los datos de los segmentos de su aplicación utilizando este grupo de registros especificado. Para obtener información general acerca de los grupos de registro, consulte Trabajar con grupos de registros y flujos de registros.
- `AWS_XRAY_TRACING_NAME`: establezca el nombre de servicio que el SDK utiliza para los segmentos. Anula el nombre de servicio que se ha establecido en la estrategia de nomenclatura de segmento del filtro de servlet.

Las variables de entorno anulan las propiedades de sistema equivalentes y los valores establecidos en el código.

Propiedades del sistema

Puede utilizar las propiedades del sistema como una alternativa específica de JVM para las variables de entorno. El SDK admite las propiedades siguientes:

- `com.amazonaws.xray.strategy.tracingName`: equivalente a `AWS_XRAY_TRACING_NAME`.
- `com.amazonaws.xray.emitters.daemonAddress`: equivalente a `AWS_XRAY_DAEMON_ADDRESS`.
- `com.amazonaws.xray.strategy.contextMissingStrategy`: equivalente a `AWS_XRAY_CONTEXT_MISSING`.

Si se ha establecido tanto una propiedad del sistema y como su variable de entorno equivalente, se utiliza el valor de la variable de entorno. En cualquier caso, esos valores anulan los valores establecidos en el código.

Rastreo de las solicitudes entrantes con el SDK de X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información

sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede usar el SDK de X-Ray para rastrear las solicitudes HTTP entrantes que su aplicación sirve en una EC2 instancia de Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Utilice un `Filter` para instrumentar las solicitudes HTTP entrantes. Cuando añade el filtro de servlet de X-Ray a su aplicación, el SDK de X-Ray para Java crea un segmento para cada solicitud muestreada. Este segmento incluye el momento, el método y la disposición de la solicitud HTTP. La instrumentación adicional crea subsegmentos en este segmento.

> **Note**
>
> Para AWS Lambda las funciones, Lambda crea un segmento para cada solicitud muestreada. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

Cada segmento tiene un nombre que identifica la aplicación en el mapa de servicio. El nombre del segmento se puede asignar de forma estática o se puede configurar el SDK para que le asigne un nombre dinámico en función del encabezado del host de la solicitud entrante. La nomenclatura dinámica permite agrupar los rastros en función del nombre de dominio de la solicitud y aplicar un nombre predeterminado si el nombre no coincide con el patrón esperado (por ejemplo, si el encabezado del host está falsificado).

> **Solicitudes reenviadas**
>
> Si un equilibrador de carga u otro intermediario reenvía una solicitud a la aplicación, X-Ray toma la IP de cliente del encabezado `X-Forwarded-For` de la solicitud en lugar de tomar la IP de origen del paquete IP. La IP de cliente que se graba para una solicitud reenviada puede estar falsificada, por lo que no se debe confiar en ella.

Cuando se reenvía una solicitud, el SDK crea un campo adicional en el segmento para indicar que se realizó esta acción. Si el segmento contiene el campo `x_forwarded_for` configurado en `true`, el IP del cliente se obtiene a partir del encabezado `X-Forwarded-For` en la solicitud HTTP.

El controlador de mensajes crea un segmento para cada solicitud entrante con un bloque `http` que contiene la siguiente información:

- Método HTTP: GET, POST, PUT, DELETE, etc.
- Dirección del cliente: la dirección IP del cliente que envió la solicitud.
- Código de respuesta: el código de respuesta HTTP para la solicitud finalizada.
- Intervalo: la hora de inicio (cuando se recibió la solicitud) y la hora de finalización (cuando se envió la respuesta).
- Agente del usuario: el `user-agent` de la solicitud.
- Longitud del contenido: la `content-length` de la respuesta.

Secciones

Agregar un filtro de seguimiento a la aplicación (Tomcat)

Para Tomcat, añada un `<filter>` al archivo `web.xml` de su proyecto. Use el parámetro `fixedName` para especificar un nombre de servicio que se aplique a los segmentos creados para las solicitudes entrantes.

Example WEB-INF/web.xml - Tomcat

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.javax.servlet.AWSXRayServletFilter</filter-class>
  <init-param>
    <param-name>fixedName</param-name>
    <param-value>MyApp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>AWSXRayServletFilter</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
```

Agregar un filtro de seguimiento a la aplicación (Spring)

Para Spring, añada un `Filter` a la clase `WebConfig`. Pase el nombre del segmento al constructor `AWSXRayServletFilter` como cadena.

Example src/main/java/myapp/WebConfig.java: primavera

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
}
```

Configuración de una estrategia de nomenclatura de segmentos

AWS X-Ray utiliza un nombre de servicio para identificar la aplicación y distinguirla del resto de aplicaciones, bases de datos, recursos externos APIs y otros AWS recursos que utiliza la aplicación. Cuando el SDK de X-Ray genera segmentos para las solicitudes entrantes, registra el nombre del servicio de la aplicación en el campo de nombre del segmento.

El SDK de X-Ray puede nombrar los segmentos utilizando el nombre de host en el encabezado de la solicitud HTTP. Sin embargo, este encabezado se puede falsificar, lo que podría provocar nodos inesperados en el mapa de servicio. Para evitar que el SDK nombre los segmentos de forma incorrecta debido a que las solicitudes tienen encabezados de host falsificados, debe especificar un nombre predeterminado para las solicitudes entrantes.

Si la aplicación atiende solicitudes de varios dominios, puede configurar el SDK para que utilice una estrategia de nomenclatura dinámica que refleje esto en los nombres de los segmentos. Una estrategia de nomenclatura dinámica permite al SDK usar el nombre de host para las solicitudes que coinciden con un patrón esperado y aplicar el nombre predeterminado a las solicitudes que no coincidan.

Por ejemplo, es posible que tenga una sola aplicación que atienda solicitudes a tres subdominios: `www.example.com`, `api.example.com` y `static.example.com`. Puede usar una estrategia de nomenclatura dinámica con el patrón `*.example.com` para identificar los segmentos de cada subdominio con un nombre diferente, lo que da como resultado tres nodos de servicio en el mapa de servicio. Si su aplicación recibe solicitudes con un nombre de host que no coincide con el patrón, verá un cuarto nodo en el mapa de servicio con el nombre alternativo que especifique.

Para utilizar el mismo nombre para todos los segmentos de solicitud, especifique el nombre de la aplicación cuando inicie el filtro de servlet, tal y como se muestra en la sección anterior. Esto tiene el mismo efecto que crear un fijo SegmentNamingStrategyllamándolo `SegmentNamingStrategy.fixed()` y pasándolo al `AWSXRayServletFilter`constructor.

> **Note**
>
> Puede anular el nombre de servicio predeterminado que ha definido en el código mediante la `AWS_XRAY_TRACING_NAME`variable de entorno???.

Una estrategia de nomenclatura dinámica define un patrón con el que deben coincidir los nombres de host y un nombre predeterminado que se utiliza si el nombre de host de la solicitud HTTP no coincide con el patrón. Para asignar nombres de forma dinámica en Tomcat, utilice `dynamicNamingRecognizedHosts` y `dynamicNamingFallbackName` para definir el patrón y el nombre predeterminado, respectivamente.

Example WEB-INF/web.xml: filtro de servlet con nomenclatura dinámica

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.javax.servlet.AWSXRayServletFilter</filter-class>
  <init-param>
    <param-name>dynamicNamingRecognizedHosts</param-name>
    <param-value>*.example.com</param-value>
  </init-param>
  <init-param>
    <param-name>dynamicNamingFallbackName</param-name>
    <param-value>MyApp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>AWSXRayServletFilter</filter-name>
  <url-pattern>*</url-pattern>
```

```
</filter-mapping>
```

Para Spring, crea una dinámica SegmentNamingStrategymediante una llamada `SegmentNamingStrategy.dynamic()` y pásala al `AWSXRayServletFilter` constructor.

Example src/main/java/myapp/WebConfig.java: filtro de servlets con nomenclatura dinámica

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.strategy.SegmentNamingStrategy;

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter(SegmentNamingStrategy.dynamic("MyApp",
 "*.example.com"));
  }
}
```

Rastreo de llamadas al AWS SDK con el X-Ray SDK for Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando la aplicación realiza llamadas Servicios de AWS para almacenar datos, escribir en una cola o enviar notificaciones, el X-Ray SDK for Java rastrea las llamadas en sentido descendente en subsegmentos. Los recursos y Servicios de AWS rastreados a los que accede dentro de dichos

servicios (por ejemplo, un bucket de Amazon S3 o una cola de Amazon SQS) aparecerán como nodos posteriores en el mapa de rastros en la consola de X-Ray.

El SDK de X-Ray para Java instrumenta automáticamente todos los clientes del SDK de AWS cuando usted incluye el `aws-sdk` y un submódulo `aws-sdk-instrumentor` en su compilación. Si no incluye el submódulo Instrumentor, puede elegir instrumentar ciertos clientes a la vez que omite otros.

Para instrumentar clientes individuales, elimina el `aws-sdk-instrumentor` submódulo de la compilación y añade uno `XRayClient` como a tu cliente del AWS SDK mediante `TracingHandler` el generador de clientes del servicio.

Por ejemplo, para instrumentar un cliente `AmazonDynamoDB`, transfiera un controlador de rastros a `AmazonDynamoDBClientBuilder`.

Example MyModel.java: cliente de DynamoDB

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.handlers.TracingHandler;

...
public class MyModel {
  private AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
        .withRegion(Regions.fromName(System.getenv("AWS_REGION")))
        .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
        .build();
...
```

Para todos los servicios, puede ver el nombre de la API a la que se llama en la consola de X-Ray. Para un subconjunto de servicios, el SDK de X-Ray agrega información al segmento para proporcionar una mayor granularidad en el mapa de servicio.

Por ejemplo, cuando realiza una llamada con un cliente instrumentado de DynamoDB, el SDK agrega el nombre de tabla al segmento para las llamadas que se dirigen a una tabla. En la consola, cada tabla aparece como nodo independiente en el mapa de servicio, con un nodo genérico de DynamoDB para las llamadas que no se dirigen a una tabla.

Example Subsegmento para una llamada a DynamoDB con el fin de guardar un elemento

```
{
  "id": "24756640c0d0978a",
```

```
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Cuando accede a recursos designados, las llamadas a los siguientes servicios crean nodos adicionales en el mapa de servicio. Las llamadas que no están dirigidas a recursos concretos crean un nodo genérico en el servicio.

- Amazon DynamoDB: nombre de tabla
- Amazon Simple Storage Service: nombre de bucket y de clave
- Amazon Simple Queue Service: nombre de cola

Para instrumentar las llamadas posteriores a la versión Servicios de AWS AWS SDK for Java 2.2 y versiones posteriores, puede omitir el `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` módulo de la configuración de compilación. Incluya `aws-xray-recorder-sdk-aws-sdk-v2 module` en su lugar y después instrumente los clientes por separado configurándolos con `TracingInterceptor`.

Example AWS SDK for Java 2.2 y versiones posteriores: interceptor de rastreo

```
import com.amazonaws.xray.interceptors.TracingInterceptor;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
//...
public class MyModel {
private DynamoDbClient client = DynamoDbClient.builder()
.region(Region.US_WEST_2)
.overrideConfiguration(ClientOverrideConfiguration.builder()
```

```
.addExecutionInterceptor(new TracingInterceptor())
.build()
)
.build();
//...
```

Rastreo de llamadas a servicios web HTTP posteriores con el SDK de X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando tu aplicación realiza llamadas a microservicios o a HTTP públicos APIs, puedes usar la versión del SDK de X-Ray para Java `HttpClient` para instrumentar esas llamadas y añadir la API al gráfico del servicio como un servicio descendente.

El SDK de X-Ray para Java incluye `DefaultHttpClient` `HttpClientBuilder` clases que se pueden usar en lugar de HttpComponents los equivalentes de Apache para instrumentar las llamadas HTTP salientes.

- `com.amazonaws.xray.proxies.apache.http.DefaultHttpClient` - `org.apache.http.impl.client.DefaultHttpClient`
- `com.amazonaws.xray.proxies.apache.http.HttpClientBuilder` - `org.apache.http.impl.client.HttpClientBuilder`

Estas bibliotecas se encuentran en el submódulo `aws-xray-recorder-sdk-apache-http`.

Puede sustituir las instrucciones actuales de importación con el equivalente a X-Ray para instrumentar todos los clientes, o bien utilizar el nombre completo cuando inicie un cliente para instrumentar clientes específicos.

Example HttpClientBuilder

```
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.util.EntityUtils;
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
  public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
    HttpGet httpGet = new HttpGet("http://names.example.com/api/");
    CloseableHttpResponse response = httpclient.execute(httpGet);
    try {
      HttpEntity entity = response.getEntity();
      InputStream inputStream = entity.getContent();
      ObjectMapper mapper = new ObjectMapper();
      Map<String, String> jsonMap = mapper.readValue(inputStream, Map.class);
      String name = jsonMap.get("name");
      EntityUtils.consume(entity);
      return name;
    } finally {
      response.close();
    }
  }
```

Cuando se instrumenta una llamada a una API web posterior, el del SDK de X-Ray para Java registra un subsegmento con información sobre la solicitud HTTP y la respuesta. X-Ray utiliza el subsegmento para generar un segmento inferido de la API remota.

Example Subsegmento para una llamada HTTP posterior

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
```

```
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example Segmento inferido para una llamada HTTP posterior

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

Rastreo de consultas SQL con el SDK de X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información

sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Interceptores SQL

Instrumente consultas de base de datos SQL añadiendo el interceptor JDBC del SDK de X-Ray para Java a la configuración del origen de datos.

- PostgreSQL – `com.amazonaws.xray.sql.postgres.TracingInterceptor`
- MySQL – `com.amazonaws.xray.sql.mysql.TracingInterceptor`

Estos interceptadores están en los submódulos `aws-xray-recorder-sql-postgres` y `aws-xray-recorder-sql-mysql`, respectivamente. Implementan `org.apache.tomcat.jdbc.pool.JdbcInterceptor` y son compatibles con grupos de conexión Tomcat.

> **Note**
>
> Los interceptores de SQL no registran la consulta SQL en subsegmentos por motivos de seguridad.

Para Spring, añada el interceptor a un archivo de propiedades y cree el origen de datos con `DataSourceBuilder` de Spring Boot.

Example **`src/main/java/resources/application.properties`**: interceptor JDBC de PostgreSQL

```
spring.datasource.continue-on-error=true
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.jdbc-interceptors=com.amazonaws.xray.sql.postgres.TracingInterceptor
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

Example **`src/main/java/myapp/WebConfig.java`**: origen de datos

```
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
```

```
import org.springframework.boot.autoconfigure.jdbc.DataSourceBuilder;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import javax.servlet.Filter;
import javax.sql.DataSource;
import java.net.URL;

@Configuration
@EnableAutoConfiguration
@EnableJpaRepositories("myapp")
public class RdsWebConfig {

  @Bean
  @ConfigurationProperties(prefix = "spring.datasource")
  public DataSource dataSource() {
      logger.info("Initializing PostgreSQL datasource");
      return DataSourceBuilder.create()
              .driverClassName("org.postgresql.Driver")
              .url("jdbc:postgresql://" + System.getenv("RDS_HOSTNAME") + ":" +
 System.getenv("RDS_PORT") + "/ebdb")
              .username(System.getenv("RDS_USERNAME"))
              .password(System.getenv("RDS_PASSWORD"))
              .build();
  }
...
}
```

Para Tomcat, llame a `setJdbcInterceptors` en el origen de datos JDBC con una referencia a la clase del SDK de X-Ray para Java.

Example **`src/main/myapp/model.java`**: origen de datos

```
import org.apache.tomcat.jdbc.pool.DataSource;
...
DataSource source = new DataSource();
source.setUrl(url);
source.setUsername(user);
source.setPassword(password);
source.setDriverClassName("com.mysql.jdbc.Driver");
source.setJdbcInterceptors("com.amazonaws.xray.sql.mysql.TracingInterceptor;");
```

El SDK de X-Ray para Java incluye la biblioteca de origen de datos JDBC Tomcat, pero puede declararla como una dependencia para documentar su uso.

Example **`pom.xml`**: origen de datos JDBC

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jdbc</artifactId>
  <version>8.0.36</version>
  <scope>provided</scope>
</dependency>
```

Decorador de rastreo SQL nativo

- Añada `aws-xray-recorder-sdk-sql` a sus dependencias.
- Decore el origen de datos, conexión o declaración de su base de datos.

```
dataSource = TracingDataSource.decorate(dataSource)
connection = TracingConnection.decorate(connection)
statement = TracingStatement.decorateStatement(statement)
preparedStatement = TracingStatement.decoratePreparedStatement(preparedStatement,
 sql)
callableStatement = TracingStatement.decorateCallableStatement(callableStatement,
 sql)
```

Generación de subsegmentos personalizados con el SDK de X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

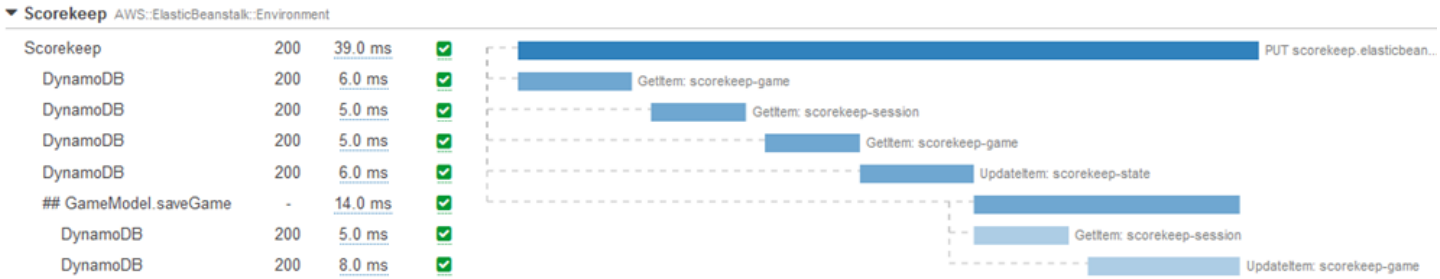
Los subsegmentos amplían el segmento de un rastro con detalles sobre el trabajo realizado para atender una solicitud. Cada vez que usted realiza una llamada con un cliente instrumentado, el SDK de X-Ray registra la información generada en un subsegmento. Puede crear subsegmentos adicionales para agrupar otros subsegmentos, medir el rendimiento de una sección de código o registrar anotaciones y metadatos.

Para administrar los subsegmentos, utilice los métodos `beginSubsegment` y `endSubsegment`.

Example GameModel.java: subsegmento personalizado

```
import com.amazonaws.xray.AWSXRay;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("Save Game");
    try {
      // check session
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
        throw new SessionNotFoundException(sessionId);
      }
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

En este ejemplo, el código del subsegmento carga la sesión del juego desde DynamoDB con un método del modelo de sesión y utiliza el mapeador de DynamoDB para AWS SDK for Java guardar la partida. Cuando se encapsula este código en un subsegmento, las llamadas a DynamoDB se convierten en elementos secundarios del subsegmento `Save Game` en la vista de rastros en la consola.



Si en el código en su subsegmentos aparecen excepciones comprobadas, envuélvalo en un bloque `try` y llame a `AWSXRay.endSubsegment()` en un bloque `finally` para garantizar que el subsegmento está siempre cerrado. Si un subsegmento no está cerrado, el segmento primario no se puede completar y no se envía a X-Ray.

En los códigos donde no aparecen excepciones comprobadas, puede transferir el código a `AWSXRay.CreateSubsegment` como una función de Lambda.

Example Función Lambda de subsegmento

```
import com.amazonaws.xray.AWSXRay;

AWSXRay.createSubsegment("getMovies", (subsegment) -> {
    // function code
});
```

Cuando crea un subsegmento dentro de un segmento o de otro subsegmento, el SDK de X-Ray para Java genera un ID para dicho subsegmento y registra la hora de inicio y la hora de finalización.

Example Subsegmentos con metadatos

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

Para la programación asíncrona y multiproceso, debe pasar manualmente el subsegmento al método `endSubsegment()` para asegurarse de que se cierra correctamente, ya que el contexto de X-Ray puede modificarse durante la ejecución asíncrona. Si un subsegmento asíncrono se cierra después de cerrar su segmento principal, este método transmitirá automáticamente todo el segmento al daemon de X-Ray.

Example Subsegmento asíncrono

```
@GetMapping("/api")
public ResponseEntity<?> api() {
  CompletableFuture.runAsync(() -> {
      Subsegment subsegment = AWSXRay.beginSubsegment("Async Work");
      try {
          Thread.sleep(3000);
      } catch (InterruptedException e) {
          subsegment.addException(e);
          throw e;
      } finally {
          AWSXRay.endSubsegment(subsegment);
      }
  });
  return ResponseEntity.ok().build();
}
```

Adición de anotaciones y metadatos a los segmentos con el SDK de X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede registrar información adicional acerca de las solicitudes, el entorno o su aplicación con anotaciones y metadatos. Puede añadir anotaciones y metadatos a los segmentos que crea el SDK de X-Ray o a los subsegmentos personalizados que cree usted mismo.

Las anotaciones son pares de clave-valor con valores de cadena, numéricos o booleanos. Las anotaciones se indexan para su uso con expresiones de filtro. Utilice anotaciones para registrar los datos que desee utilizar para agrupar rastros en la consola o cuando llame a la API de `GetTraceSummaries`.

Los metadatos son pares de clave-valor con valores de cualquier tipo, por ejemplo objetos y listas, pero que no se indexan para utilizarlos con expresiones de filtro. Utilice los metadatos para registrar datos adicionales que desee almacenar en el rastro, pero que no necesite usar para hacer búsquedas.

Además de anotaciones y metadatos, también puede registrar cadenas de ID de usuario en los segmentos. IDs Los usuarios se registran en un campo independiente en los segmentos y se indexan para usarlos en la búsqueda.

Secciones

- Registro de anotaciones con el SDK de X-Ray para Java
- Registro de metadatos con el SDK de X-Ray para Java
- Grabación del usuario IDs con el SDK de X-Ray para Java

Registro de anotaciones con el SDK de X-Ray para Java

Utilice anotaciones para registrar información sobre segmentos o subsegmentos que desee indexar para las búsquedas.

Requisitos de anotación

- Claves: la clave de una anotación de X-Ray puede contener hasta 500 caracteres alfanuméricos. No se pueden usar espacios ni símbolos, excepto el punto (.)
- Valores: el valor de una anotación de X-Ray puede contener hasta 1000 caracteres Unicode.
- Número de anotaciones: se pueden utilizar hasta 50 anotaciones por rastro.

Para registrar anotaciones

1. Obtenga una referencia al segmento o subsegmento actual desde `AWSXRay`.

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
...
Segment document = AWSXRay.getCurrentSegment();
```

o

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Subsegment;
...
Subsegment document = AWSXRay.getCurrentSubsegment();
```

2. Llame a `putAnnotation` con una clave de cadena y un valor booleano, numérico o de cadena.

```
document.putAnnotation("mykey", "my value");
```

El siguiente ejemplo muestra cómo llamar a `putAnnotation` con una clave de cadena que incluye un punto y un valor booleano, numérico o de cadena.

```
document.putAnnotation("testkey.test", "my value");
```

El SDK registra las anotaciones como pares de clave-valor en un objeto `annotations` del documento de segmento. Si llama dos veces a `putAnnotation` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Para encontrar rastros que tengan anotaciones con valores específicos, utilice la palabra clave `annotation[key]` en una expresión de filtro.

Example **src/main/java/scorekeep/GameModel.java**: anotaciones y metadatos

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## GameModel.saveGame");
    try {
      // check session
```

```
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
        throw new SessionNotFoundException(sessionId);
      }
      Segment segment = AWSXRay.getCurrentSegment();
      subsegment.putMetadata("resources", "game", game);
      segment.putAnnotation("gameid", game.getId());
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

Registro de metadatos con el SDK de X-Ray para Java

Utilice los metadatos para registrar información sobre segmentos o subsegmentos que no necesite indexar para las búsquedas. Los valores de metadatos pueden ser cadenas, números, booleanos o cualquier objeto que se pueda serializar en un objeto o matriz JSON.

Para registrar metadatos

1. Obtenga una referencia al segmento o subsegmento actual desde `AWSXRay`.

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

   o

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Subsegment;
   ...
   Subsegment document = AWSXRay.getCurrentSubsegment();
   ```

2. Llame a `putMetadata` con un espacio de nombres de cadena, una clave de cadena y un valor booleano, numérico, de cadena o de objeto.

```
document.putMetadata("my namespace", "my key", "my value");
```

o

Llame a `putMetadata` con solo una clave y un valor.

```
document.putMetadata("my key", "my value");
```

Si no especifica un espacio de nombres, el SDK utiliza `default`. Si llama dos veces a `putMetadata` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Example **src/main/java/scorekeep/GameModel.java**: anotaciones y metadatos

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## GameModel.saveGame");
    try {
      // check session
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
        throw new SessionNotFoundException(sessionId);
      }
      Segment segment = AWSXRay.getCurrentSegment();
      subsegment.putMetadata("resources", "game", game);
      segment.putAnnotation("gameid", game.getId());
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

Grabación del usuario IDs con el SDK de X-Ray para Java

Registre IDs el usuario en los segmentos de solicitud para identificar al usuario que envió la solicitud.

Para registrar al usuario IDs

1. Obtenga una referencia al segmento actual desde `AWSXRay`.

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
...
Segment document = AWSXRay.getCurrentSegment();
```

2. Llame a `setUser` con un ID de cadena del usuario que envió la solicitud.

```
document.setUser("U12345");
```

Puede llamar a `setUser` en sus controladores para registrar el ID de usuario en cuanto la aplicación empiece a procesar la solicitud. Si solo va a utilizar el segmento para establecer el ID de usuario, puede encadenar las llamadas en una sola línea.

Example src/main/java/scorekeep/MoveController.java: ID de usuario

```
import com.amazonaws.xray.AWSXRay;
...
  @RequestMapping(value="/{userId}", method=RequestMethod.POST)
  public Move newMove(@PathVariable String sessionId, @PathVariable String
 gameId, @PathVariable String userId, @RequestBody String move) throws
 SessionNotFoundException, GameNotFoundException, StateNotFoundException,
 RulesException {
    AWSXRay.getCurrentSegment().setUser(userId);
    return moveFactory.newMove(sessionId, gameId, userId, move);
  }
```

Para buscar rastros de un ID de usuario, utilice la palabra clave `user` en una expresión de filtro.

AWS X-Ray métricas del X-Ray SDK para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

En este tema se describen el espacio de AWS X-Ray nombres, las métricas y las dimensiones. Puedes usar el SDK de X-Ray para Java para publicar CloudWatch métricas de Amazon sin muestrear de los segmentos de X-Ray recopilados. Estas métricas se derivan de la horade inicio y finalización del segmento, y los marcadores de estado limitado, fallo y error. Utilice estas métricas de seguimiento para exponer los reintentos y los problemas de dependencia con los subsegmentos.

CloudWatch es un repositorio de métricas. Una métrica es el concepto fundamental CloudWatch y representa un conjunto de puntos de datos ordenados en el tiempo. Usted (o Servicios de AWS) publica puntos de datos de métricas CloudWatch y recupera las estadísticas sobre esos puntos de datos como un conjunto ordenado de datos de series temporales.

Las métricas se definen de forma exclusiva mediante un nombre, un espacio de nombres y una o varias dimensiones. Cada punto de datos tiene una marca temporal y, opcionalmente, una unidad de medida. Cuando se solicitan estadísticas, el flujo de datos devuelto se identifica mediante el espacio de nombres, el nombre de la métrica y la dimensión.

Para obtener más información al respecto CloudWatch, consulta la Guía del CloudWatch usuario de Amazon.

CloudWatch Métricas de X-Ray

El espacio de nombres de `ServiceMetrics/SDK` incluye las siguientes métricas.

Métrica	Estadísticas disponibles	Description (Descripción)	Unidades
`Latency`	Media, mínima, máxima, recuento	La diferencia entre la hora de inicio y finalización. Media, mínima y máxima describen la latencia operativa. Recuento describe el recuento de llamadas.	Milisegundos
`ErrorRate`	Media, Suma	La tasa de solicitudes que fallaron con un código de estado `4xx Client Error`, lo que da lugar a un error.	Porcentaje
`FaultRate`	Media, Suma	La tasa de seguimientos que fallaron con un código de estado `5xx Server Error`, lo que da lugar a un fallo.	Porcentaje
`ThrottleRate`	Media, Suma	La tasa de registros limitas que devuelven un código de estado `429`. Se trata de un subconjunto de la métrica `ErrorRate` .	Porcentaje
`OkRate`	Media, Suma	La tasa de solicitudes rastreadas que dan lugar a un código de estado `OK`.	Porcentaje

CloudWatch Dimensiones de X-Ray

Utilice las dimensiones de la siguiente tabla para ajustar las métricas devueltas para sus aplicaciones de Java instrumentadas de X-Ray.

Dimensión	Description (Descripción)
`ServiceType`	El tipo de servicio, por ejemplo, `AWS::EC2::Instance` o `NONE`, si no se conoce.
`ServiceName`	El nombre canónico del servicio.

Habilitar las CloudWatch métricas de X-Ray

Utilice el siguiente procedimiento para habilitar métricas de rastreo en su aplicación de Java instrumentada.

Para configurar las métricas de seguimiento

1. Agregue el paquete `aws-xray-recorder-sdk-metrics` como una dependencia de Apache Maven. Para obtener más información, consulte Submódulos del SDK de X-Ray para Java.
2. Habilite un nuevo `MetricsSegmentListener()` como parte de la compilación de la grabadora global.

   Example src/com/myapp/web/Startup.java

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.AWSXRayRecorderBuilder;
   import com.amazonaws.xray.plugins.EC2Plugin;
   import com.amazonaws.xray.plugins.ElasticBeanstalkPlugin;
   import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

   @Configuration
   public class WebConfig {
   ...
     static {
       AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
                                          .standard()
                                          .withPlugin(new EC2Plugin())
                                          .withPlugin(new ElasticBeanstalkPlugin())
   ```

```
                                        .withSegmentListener(new
 MetricsSegmentListener());

    URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
    builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

    AWSXRay.setGlobalRecorder(builder.build());
  }
}
```

3. Implemente el CloudWatch agente para recopilar métricas mediante Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS) o Amazon Elastic Kubernetes Service (Amazon EKS):

   - Para configurar Amazon EC2, consulte Instalación del CloudWatch agente.
   - Para configurar Amazon ECS, consulte Supervisión de los contenedores de Amazon ECS mediante Información de contenedores.
   - Para configurar Amazon EKS, consulte Instalación del CloudWatch agente mediante el complemento Amazon CloudWatch Observability EKS.

4. Configure el SDK para que se comunique con el CloudWatch agente. De forma predeterminada, el SDK se comunica con el CloudWatch agente en la dirección`127.0.0.1`. Puede configurar direcciones alternativas al configurar el entorno variable o la propiedad de Java en `address:port`.

   Example Variable de entorno

   ```
   AWS_XRAY_METRICS_DAEMON_ADDRESS=address:port
   ```

   Example Propiedad de Java

   ```
   com.amazonaws.xray.metrics.daemonAddress=address:port
   ```

Para validar la configuración

1. Inicie sesión en Consola de administración de AWS y abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. Abra la pestaña Metrics (Métricas) para observar el flujo de sus métricas.

3. (Opcional) En la CloudWatch consola, en la pestaña Registros, abra el grupo de `ServiceMetricsSDK` registros. Busque un flujo de registros que coincida con las métricas del host y confirme los mensajes del registro.

Transmisión de contexto de segmento entre subprocesos en una aplicación multiproceso

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando se crea un nuevo subproceso en la aplicación, el `AWSXRayRecorder` no mantiene una referencia al segmento o subsegmento Entity actual. Si utilizas un cliente instrumentado en el nuevo hilo, el SDK intenta escribir en un segmento que no existe, lo que provoca un. SegmentNotFoundException

Para evitar que se produzcan excepciones durante el desarrollo, puedes configurar la grabadora con una ContextMissingStrategyque le indique que registre un error en su lugar. Puede configurar la estrategia en código con SetContextMissingStrategy, o configurar opciones equivalentes con, una variable de entorno o una propiedad del sistema.

Una forma de abordar el error consiste en utilizar un segmento nuevo llamando a beginSegment al iniciar el subproceso y a endSegment al cerrarlo. Esto funciona si está instrumentando código que no se ejecuta en respuesta a una solicitud HTTP, como código que se ejecuta cuando se inicia la aplicación.

Si utiliza varios subprocesos para gestionar las solicitudes entrantes, puede transferir el segmento o subsegmento actual al nuevo subproceso y facilitarlo a la grabadora global. De este modo, se garantiza que la información registrada en el nuevo subproceso esté asociada al mismo segmento que el resto de la información registrada sobre dicha solicitud. Una vez que el segmento esté

disponible en el nuevo hilo, puede ejecutar cualquier ejecutable con acceso al contexto de ese segmento mediante el método `segment.run(() -> { ... })`.

Consulte Uso de clientes instrumentados en subprocesos de trabajo para ver un ejemplo.

Uso de X-Ray con programación asíncrona

El SDK de X-Ray para Java se puede utilizar en programas Java asíncronos con. SegmentContextExecutors SegmentContextExecutor Implementa la interfaz Executor, lo que significa que se puede transferir a todas las operaciones asíncronas de un. CompletableFuture Eso garantiza que cualquier operación asíncrona se ejecute con el segmento correcto en su contexto.

Example Ejemplo: App.java: Pasando a SegmentContextExecutor CompletableFuture

```
DynamoDbAsyncClient client = DynamoDbAsyncClient.create();

AWSXRay.beginSegment();

// ...

client.getItem(request).thenComposeAsync(response -> {
    // If we did not provide the segment context executor, this request would not be
 traced correctly.
    return client.getItem(request2);
}, SegmentContextExecutors.newSegmentContextExecutor());
```

AOP con Spring y el SDK de X-Ray para Java

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

En este tema se describe cómo utilizar el SDK de X-Ray y el marco de trabajo Spring para instrumentar una aplicación sin cambiar su lógica básica. Esto significa que ahora existe una forma no invasiva de instrumentar las aplicaciones que se ejecutan de forma remota. AWS

Para habilitar AOP en Spring

1. Configure Spring
2. Añada un filtro de rastreo a la aplicación.
3. Comente el código o implemente una interfaz
4. Active X-Ray en la aplicación

Configuración de Spring

Puede utilizar Maven o Gradle para configurar Spring para que utilice AOP con el fin de instrumentar la aplicación.

Si utiliza Maven para compilar la aplicación, añada la siguiente dependencia al archivo `pom.xml`.

```
<dependency>
     <groupId>com.amazonaws</groupId>
     <artifactId>aws-xray-recorder-sdk-spring</artifactId>
     <version>2.11.0</version>
</dependency>
```

Para Gradle, añada la siguiente dependencia al archivo `build.gradle`.

```
compile 'com.amazonaws:aws-xray-recorder-sdk-spring:2.11.0'
```

Configuración de Spring Boot

Además de la dependencia de Spring descrita en la sección anterior, si utiliza Spring Boot, añada la siguiente dependencia si aún no está en su classpath.

Maven:

```
<dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-aop</artifactId>
     <version>2.5.2</version>
```

```
</dependency>
```

Gradle:

```
compile 'org.springframework.boot:spring-boot-starter-aop:2.5.2'
```

Adición de un filtro de rastreo a la aplicación

Añada un `Filter` a su clase `WebConfig`. Pase el nombre del segmento al constructor `AWSXRayServletFilter` como cadena. Para obtener más información sobre los filtros de rastreo y la instrumentación de las solicitudes entrantes, consulte Rastreo de las solicitudes entrantes con el SDK de X-Ray para Java.

Example src/main/java/myapp/WebConfig.java: primavera

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
}
```

Compatibilidad con Jakarta

Spring 6 usa Jakarta en lugar de Javax para su Enterprise Edition. Para admitir este nuevo espacio de nombres, X-Ray ha creado un conjunto paralelo de clases que viven en su propio espacio de nombres de Yakarta.

Para las clases de filtro, sustituya `javax` por `jakarta`. Al configurar una estrategia de nomenclatura de segmentos, agregue `jakarta` antes del nombre de la clase de estrategia de nomenclatura, como en el siguiente ejemplo:

```
package myapp;
```

```
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import jakarta.servlet.Filter;
import com.amazonaws.xray.jakarta.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.strategy.jakarta.SegmentNamingStrategy;

@Configuration
public class WebConfig {
    @Bean
    public Filter TracingFilter() {
        return new AWSXRayServletFilter(SegmentNamingStrategy.dynamic("Scorekeep"));
    }
}
```

Anotación del código o implementación de una interfaz

Las clases deben anotarse con la anotación `@XRayEnabled` o deben implementar la interfaz de `XRayTraced`. Esto indica al sistema de AOP que encapsule las funciones de la clase afectada para la instrumentación de X-Ray.

Activación de X-Ray en la aplicación

Para activar el rastreo de X-Ray en la aplicación, el código debe extender la clase abstracta `BaseAbstractXRayInterceptor` anulando los siguientes métodos.

- `generateMetadata`: esta función permite la personalización de los metadatos adjuntados al rastreo de la función actual. De forma predeterminada, el nombre de clase de la función que se ejecuta se registra en los metadatos. Puede añadir más datos si necesita más información.
- `xrayEnabledClasses`: esta función está vacía, y debe seguir así. Sirve para alojar un conjunto de puntos de unión (pointcut) que indica al interceptor los métodos que debe encapsular. Defina el pointcut especificando las clases que se han anotado con `@XRayEnabled` que se deben rastrear. La siguiente instrucción pointcut indica al interceptor que encapsule todos los beans del controlador anotados con la anotación `@XRayEnabled`.

  ```
  @Pointcut(“@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)”)
  ```

Si su proyecto utiliza Spring Data JPA, considere la posibilidad de ampliar desde `AbstractXRayInterceptor` en lugar de `BaseAbstractXRayInterceptor`.

Ejemplo

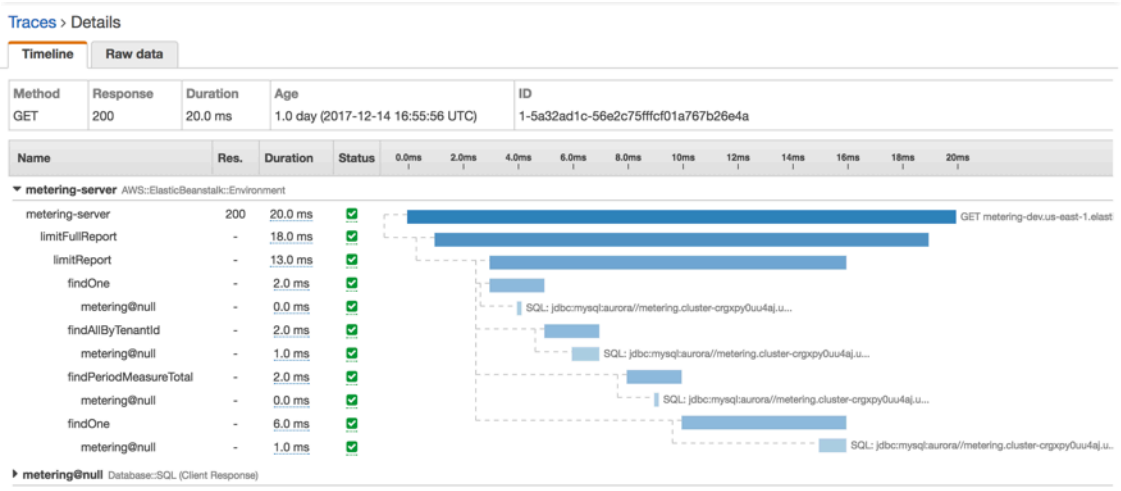
El siguiente código extiende la clase abstracta `BaseAbstractXRayInterceptor`.

```
@Aspect
@Component
public class XRayInspector extends BaseAbstractXRayInterceptor {
    @Override
    protected Map<String, Map<String, Object>> generateMetadata(ProceedingJoinPoint
 proceedingJoinPoint, Subsegment subsegment) throws Exception {
        return super.generateMetadata(proceedingJoinPoint, subsegment);
    }

  @Override
  @Pointcut("@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)")

  public void xrayEnabledClasses() {}

}
```

El siguiente código es una clase que X-Ray va a instrumentar.

```
@Service
@XRayEnabled
public class MyServiceImpl implements MyService {
    private final MyEntityRepository myEntityRepository;

    @Autowired
    public MyServiceImpl(MyEntityRepository myEntityRepository) {
        this.myEntityRepository = myEntityRepository;
    }

    @Transactional(readOnly = true)
    public List<MyEntity> getMyEntities(){
        try(Stream<MyEntity> entityStream = this.myEntityRepository.streamAll()){

            return entityStream.sorted().collect(Collectors.toList());
        }
    }
}
```

Si ha configurado correctamente la aplicación, debe ver la pila de llamadas completa de la aplicación, desde el controlador hasta las llamadas a los servicios, tal y como se muestra en la siguiente captura de pantalla de la consola.



Uso de Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede instrumentar su aplicación Node.js de dos maneras para enviar rastros a X-Ray:

- AWS Distro for OpenTelemetry JavaScript: una AWS distribución que proporciona un conjunto de bibliotecas de código abierto para enviar métricas y rastreos correlacionados a varias soluciones de AWS monitoreo, incluidas Amazon y Amazon OpenSearch Service CloudWatch AWS X-Ray, a través de AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK para Node.js: conjunto de bibliotecas para generar y enviar trazas a X-Ray mediante el daemon X-Ray.

Para obtener más información, consulte Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs.

AWS Distro para OpenTelemetry JavaScript

Con AWS Distro para OpenTelemetry (ADOT) JavaScript, puede instrumentar sus aplicaciones una vez y enviar métricas y rastros correlacionados a varias soluciones de supervisión de AWS, incluidas Amazon CloudWatch, AWS X-Ray y Amazon OpenSearch Service. El uso de X-Ray con AWS Distro para OpenTelemetry requiere dos componentes: un SDK de OpenTelemetry habilitado para usarlo con X-Ray y el AWS Distro para OpenTelemetry Collector habilitado para usarlo con X-Ray.

Para empezar, consulte la documentación de AWS Distro para OpenTelemetry JavaScript.

> **Note**
>
> ADOT JavaScript es compatible con todas las aplicaciones Node.js del lado del servidor. ADOT JavaScript no puede exportar datos a X-Ray desde los clientes del navegador.

Para obtener más información sobre el uso de AWS Distro para OpenTelemetry con AWS X-Ray y otros Servicios de AWS, consulte AWS Distro para OpenTelemetry o la documentación de AWS Distro para OpenTelemetry.

Para obtener información adicional sobre los lenguajes compatibles y el uso, consulte AWS Observability en GitHub.

AWS SDK de X-Ray para Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Node.js es una biblioteca para las aplicaciones web de Express y funciones de Lambda de Node.js que proporciona clases y métodos para generar y enviar datos de rastreo al daemon de X-Ray. Los datos de rastreo incluyen información sobre las solicitudes HTTP entrantes atendidas por la aplicación y las llamadas que la aplicación realiza a los servicios descendentes mediante el SDK o los AWS clientes HTTP.

> **Note**
>
> El SDK de X-Ray para Node.js es un proyecto de código abierto compatible con las versiones 14.x y posteriores de Node.js. Puedes seguir el proyecto y enviar las incidencias y solicitudes de cambios en GitHub: github. com/aws/aws- xray-sdk-node

Si usa Express, empiece agregando el SDK como middleware en el servidor de aplicaciones para rastrear solicitudes entrantes. El middleware crea un segmento para cada solicitud rastreada y lo completa cuando se envía la respuesta. Mientras el segmento está abierto, puede utilizar los métodos del cliente del SDK para añadir información al segmento y crear subsegmentos para rastrear llamadas posteriores. El SDK también registra automáticamente las excepciones que produce su aplicación mientras el segmento está abierto.

En el caso de las funciones de Lambda llamadas por una aplicación o un servicio instrumentados, Lambda lee el encabezado de rastreo y rastrea automáticamente las solicitudes muestreadas. Para otras funciones, puede configurar Lambda con el fin de muestrear y rastrear las solicitudes entrantes. En cualquier caso, Lambda crea el segmento y se lo proporciona al SDK de X-Ray.

> **Note**
>
> En Lambda, el SDK de X-Ray es opcional. Si no lo usa en su función, el mapa de servicio seguirá incluyendo un nodo para el servicio de Lambda y uno para cada función de Lambda. Al añadir el SDK, puede instrumentar el código de función para añadir subsegmentos al segmento de función registrado por Lambda. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

A continuación, utilice el SDK de X-Ray para Node.js a fin de instrumentar su AWS SDK para JavaScript los clientes de Node.js. Cada vez que realizas una llamada a un canal intermedio Servicio de AWS o a un recurso con un cliente instrumentado, el SDK registra la información sobre la llamada en un subsegmento. Servicios de AWS y los recursos a los que accedes desde los servicios aparecen como nodos descendentes en el mapa de rastreo para ayudarte a identificar los errores y los problemas de limitación en las conexiones individuales.

El SDK de X-Ray para Node.js también proporciona instrumentación para llamadas posteriores a consultas HTTP web APIs y SQL. Incluya su cliente HTTP en el método de captura del SDK para registrar información sobre las llamadas HTTP salientes. Para los clientes SQL, utilice el método de captura para el tipo de base de datos.

El middleware aplica reglas de muestreo a las solicitudes entrantes para determinar qué solicitudes se deben rastrear. Puede configurar el SDK de X-Ray para Node.js para ajustar el comportamiento del muestreo o para registrar información sobre los recursos AWS informáticos en los que se ejecuta la aplicación.

Registre información adicional acerca de las solicitudes y el trabajo que la aplicación realiza en anotaciones y metadatos. Las anotaciones son pares sencillos de clave-valor que se indexan para su uso con expresiones de filtro para poder buscar rastros que contengan datos específicos. Las entradas de metadatos son menos restrictivas y pueden registrar objetos y matrices completos, es decir, todo lo que se pueda serializar en JSON.

> **Anotaciones y metadatos**
>
> Las anotaciones y los metadatos son texto arbitrario que se agrega a los segmentos con el SDK de X-Ray. Las anotaciones se indexan para su uso con expresiones de filtro. Los metadatos no se indexan pero se pueden ver en el segmento sin procesar con la consola o la API de X-Ray. Cualquier persona a la que conceda acceso de lectura a X-Ray puede ver estos datos.

Cuando tenga muchos clientes instrumentados en su código, un único segmento de solicitud puede contener un gran número de subsegmentos, uno para cada llamada realizada con un cliente instrumentado. Puede organizar y agrupar los subsegmentos incluyendo las llamadas del cliente en subsegmentos personalizados. Puede crear un subsegmento personalizado para una función completa o para cualquier sección de código, y registrar los metadatos y las anotaciones en el subsegmento en lugar de escribirlo todo en el segmento principal.

Para tener acceso a los documentos de referencia sobre las clases y los métodos del SDK, consulte la Referencia de la API del SDK de AWS X-Ray para Node.js.

Requisitos

El SDK de X-Ray para Node.js requiere Node.js y las siguientes bibliotecas:

- `atomic-batcher`: 1.0.2
- `cls-hooked`: 4.2.2
- `pkginfo`: 0.4.0
- `semver`: 5.3.0

El SDK obtiene estas bibliotecas cuando se instala con NPM.

Para rastrear los clientes del AWS SDK, el SDK de X-Ray para Node.js requiere una versión mínima del AWS SDK para JavaScript Node.js.

- `aws-sdk`: 2.7.15

Administración de dependencias

El SDK de X-Ray para Node.js está disponible en NPM.

- Paquete: `aws-xray-sdk`

Para el desarrollo local, instale el SDK en el directorio del proyecto con npm.

```
~/nodejs-xray$ npm install aws-xray-sdk
aws-xray-sdk@3.3.3
  ### aws-xray-sdk-core@3.3.3
  # ### @aws-sdk/service-error-classification@3.15.0
  # ### @aws-sdk/types@3.15.0
  # ### @types/cls-hooked@4.3.3
  # # ### @types/node@15.3.0
  # ### atomic-batcher@1.0.2
  # ### cls-hooked@4.2.2
  # # ### async-hook-jl@1.7.6
  # # # ### stack-chain@1.3.7
  # # ### emitter-listener@1.1.2
  # #   ### shimmer@1.2.1
  # ### semver@5.7.1
  ### aws-xray-sdk-express@3.3.3
  ### aws-xray-sdk-mysql@3.3.3
  ### aws-xray-sdk-postgres@3.3.3
```

Use la opción `--save` para guardar el SDK como una dependencia en el archivo `package.json` de la aplicación.

```
~/nodejs-xray$ npm install aws-xray-sdk --save
aws-xray-sdk@3.3.3
```

Si la aplicación tiene dependencias cuyas versiones entran en conflicto con las del SDK de X-Ray, se instalarán ambas versiones para garantizar la compatibilidad. Para obtener más detalles, consulte la documentación oficial de NPM para la resolución de dependencias.

Ejemplos de Node.js

Usa el AWS X-Ray SDK de Node.js para end-to-end ver las solicitudes a medida que se desplazan por tus aplicaciones de Node.js.

- La aplicación de ejemplo Node.js está activada GitHub.

Configuración del SDK de X-Ray para Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede configurar el SDK de X-Ray para Node.js el uso de complementos a fin de incluir información sobre el servicio que sus aplicaciones ejecutan, modificar la conducta predeterminada de muestreo o agregar reglas de muestreo que se aplican a las solicitudes dirigidas a rutas específicas.

Secciones

- Complementos del servicio
- Reglas de muestreo
- Registro
- Dirección del daemon de X-Ray
- Variables de entorno

Complementos del servicio

Utilice `plugins` para registrar información sobre el servicio que aloja la aplicación.

Plugins

- Amazon EC2 : `EC2Plugin` agrega el ID de la instancia, la zona de disponibilidad y el grupo de CloudWatch registros.
- Elastic Beanstalk: `ElasticBeanstalkPlugin` añade el nombre de entorno, la etiqueta de versión y el ID de implementación.
- Amazon ECS: `ECSPlugin` agrega el ID de contenedor.

Para utilizar un complemento, configure el cliente del SDK de X-Ray para Node.js mediante el método `config`.

Example app.js: complementos

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.config([AWSXRay.plugins.EC2Plugin,AWSXRay.plugins.ElasticBeanstalkPlugin]);
```

El SDK también usa la configuración del complemento para establecer el campo `origin` en el segmento. Esto indica el tipo de AWS recurso que ejecuta la aplicación. Cuando utilizas varios complementos, el SDK utiliza el siguiente orden de resolución para determinar el origen: ElasticBeanstalk > EKS > ECS > EC2.

Reglas de muestreo

El SDK utiliza las reglas de muestreo que define el usuario en la consola de X-Ray para determinar qué solicitudes registrar. La regla predeterminada rastrea la primera solicitud cada segundo y el 5 % de las solicitudes adicionales de todos los servicios que envían rastros a X-Ray. Cree reglas adicionales en la consola de X-Ray para personalizar la cantidad de datos registrados para cada una de sus aplicaciones.

El SDK aplica las reglas personalizadas en el orden en que se definen. Si una solicitud coincide con varias reglas personalizadas, el SDK solo aplica la primera regla.

> **Note**
>
> Si el SDK no puede comunicarse con X-Ray para obtener las reglas de muestreo, vuelve a una regla local predeterminada de la primera solicitud cada segundo y del 5 % de las solicitudes adicionales por host. Esto puede ocurrir si el host no tiene permiso para llamar al

muestreo APIs o no puede conectarse al daemon X-Ray, que actúa como proxy TCP para las llamadas a la API realizadas por el SDK.

También puede configurar el SDK para que cargue las reglas de muestreo desde un documento JSON. El SDK puede usar las reglas locales como respaldo para los casos en que el muestreo de X-Ray no esté disponible, o puede usar las reglas locales exclusivamente.

Example sampling-rules.json

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

En este ejemplo se define una regla personalizada y una regla predeterminada. La regla personalizada aplica un porcentaje de muestreo del 5 % sin un número mínimo de solicitudes de rastreo para las rutas incluidas bajo `/api/move/`. La regla predeterminada rastrea la primera solicitud cada segundo y el 10 % de las solicitudes adicionales.

La desventaja de definir las reglas localmente es que el objetivo establecido lo aplica cada instancia de la grabadora de forma independiente, en lugar de ser administrado por el servicio de X-Ray. A medida que se implementan más hosts, el porcentaje establecido se multiplica, lo que dificulta el control de la cantidad de datos registrados.

Sí AWS Lambda, no puedes modificar la frecuencia de muestreo. Si un servicio instrumentado llama a su función, Lambda registrará las llamadas que generaron solicitudes muestreadas por ese

servicio. Si el rastreo activo está activado y no hay ningún encabezado de rastreo, Lambda toma la decisión de muestreo.

Para configurar reglas de respaldo, indique al SDK de X-Ray para Node.js que cargue reglas de muestreo desde un archivo con `setSamplingRules`.

Example app.js: reglas de muestreo de un archivo

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
```

También puede definir las reglas en código y pasarlas a `setSamplingRules` como un objeto.

Example app.js: reglas de muestreo de un objeto

```
var AWSXRay = require('aws-xray-sdk');
var rules = {
  "rules": [ { "description": "Player moves.", "service_name": "*", "http_method": "*",
 "url_path": "/api/move/*", "fixed_target": 0, "rate": 0.05 } ],
  "default": { "fixed_target": 1, "rate": 0.1 },
  "version": 1
  }

AWSXRay.middleware.setSamplingRules(rules);
```

Para utilizar solo reglas locales, llame a `disableCentralizedSampling`.

```
AWSXRay.middleware.disableCentralizedSampling()
```

Registro

Para registrar la salida del SDK, llame a `AWSXRay.setLogger(logger)`, donde `logger` es un objeto que proporciona métodos de registro estándar (`warn`, `info`, etc.).

De forma predeterminada, el SDK registrará los mensajes de error en la consola mediante los métodos estándar del objeto de la consola. El nivel de registro del registrador integrado se puede configurar mediante las variables de entorno `AWS_XRAY_DEBUG_MODE` o `AWS_XRAY_LOG_LEVEL`. Para obtener una lista de valores de nivel de registro válidos, consulte Variables de entorno.

Si desea proporcionar un formato o destino diferente para los registros, puede proporcionar al SDK su propia implementación de la interfaz del registrador, como se muestra a continuación. Se puede

utilizar cualquier objeto que implemente esta interfaz. Eso significa que muchas bibliotecas de registro, por ejemplo, Winston, podrían usarse y pasarse directamente al SDK.

Example app.js: registro

```
var AWSXRay = require('aws-xray-sdk');

// Create your own logger, or instantiate one using a library.
var logger = {
  error: (message, meta) => { /* logging code */ },
  warn: (message, meta) => { /* logging code */ },
  info: (message, meta) => { /* logging code */ },
  debug: (message, meta) => { /* logging code */ }
}

AWSXRay.setLogger(logger);
AWSXRay.config([AWSXRay.plugins.EC2Plugin]);
```

Llame a `setLogger` antes de ejecutar otros métodos de configuración, con el fin de asegurarse de capturar la salida de estas operaciones.

Dirección del daemon de X-Ray

Si el daemon de X-Ray escucha en un puerto o host que no sea `127.0.0.1:2000`, puede configurar el SDK de X-Ray para Node.js con el fin de enviar datos de rastreo a otra dirección.

```
AWSXRay.setDaemonAddress('host:port');
```

Puede especificar el host por nombre o IPv4 dirección.

Example app.js: dirección del demonio

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('daemonhost:8082');
```

Si ha configurado el demonio para que escuche en diferentes puertos para TCP y UDP, puede especificar ambos en la configuración de dirección del demonio.

Example app.js: dirección del demonio en puertos independientes

```
var AWSXRay = require('aws-xray-sdk');
```

```
AWSXRay.setDaemonAddress('tcp:daemonhost:8082 udp:daemonhost:8083');
```

También puede establecer la dirección del demonio utilizando la `AWS_XRAY_DAEMON_ADDRESS`variable de entorno???.

Variables de entorno

Puede usar variables de entorno para configurar el SDK de X-Ray para Node.js. El SDK admite las siguientes variables.

- `AWS_XRAY_CONTEXT_MISSING`: establezca esta opción en `RUNTIME_ERROR` para generar excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.

  Valores válidos

  - `RUNTIME_ERROR`: lance una excepción de tiempo de ejecución.
  - `LOG_ERROR`: registre un error y continúe (predeterminado).
  - `IGNORE_ERROR`: ignore el error y continúe.

  Se pueden producir errores relativos a segmentos o subsegmentos inexistentes al intentar usar un cliente instrumentado en el código de inicio que se ejecuta cuando no hay ninguna solicitud abierta o en el código que inicia un nuevo subproceso.
- `AWS_XRAY_DAEMON_ADDRESS`: establezca el host y el puerto del oyente del daemon de X-Ray. De forma predeterminada, el SDK utiliza `127.0.0.1:2000` tanto para los datos de rastro (UDP) como para el muestreo (TCP). Use esta variable si ha configurado el daemon para que escuche en un puerto diferente o si se ejecuta en un host diferente.

  Formato

  - El mismo puerto: *`address`*`:`*`port`*
  - Puertos diferentes: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_DEBUG_MODE`: establezca este valor en `TRUE` para configurar el SDK de manera que envíe los registros a la consola, a nivel de `debug`.
- `AWS_XRAY_LOG_LEVEL` : establezca un nivel de registro para el registrador predeterminado. Los valores válidos son `debug`, `info`, `warn`, `error` y `silent`. Este valor se ignora cuando AWS_XRAY_DEBUG_MODE se establece en`TRUE`.

- `AWS_XRAY_TRACING_NAME`: establezca el nombre de servicio que el SDK utiliza para los segmentos. Anula el nombre de segmento que se ha establecido en el middleware de Express.

Rastreo de las solicitudes entrantes con el SDK de X-Ray para Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede usar el SDK de X-Ray para Node.js para rastrear las solicitudes HTTP entrantes que sus aplicaciones Express y Restify sirven en una EC2 instancia de Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

El SDK de X-Ray para Node.js proporciona middleware para aplicaciones que utilizan los marcos de trabajo Express y Restify. Cuando añade el middleware de X-Ray a su aplicación, el SDK de X-Ray para Node.js crea un segmento para cada solicitud muestreada. Este segmento incluye el momento, el método y la disposición de la solicitud HTTP. La instrumentación adicional crea subsegmentos en este segmento.

> **Note**
>
> Para AWS Lambda las funciones, Lambda crea un segmento para cada solicitud muestreada. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

Cada segmento tiene un nombre que identifica la aplicación en el mapa de servicio. El nombre del segmento se puede asignar de forma estática o se puede configurar el SDK para que le asigne un nombre dinámico en función del encabezado del host de la solicitud entrante. La nomenclatura dinámica permite agrupar los rastros en función del nombre de dominio de la solicitud y aplicar

un nombre predeterminado si el nombre no coincide con el patrón esperado (por ejemplo, si el encabezado del host está falsificado).

> **Solicitudes reenviadas**
>
> Si un equilibrador de carga u otro intermediario reenvía una solicitud a la aplicación, X-Ray toma la IP de cliente del encabezado `X-Forwarded-For` de la solicitud en lugar de tomar la IP de origen del paquete IP. La IP de cliente que se graba para una solicitud reenviada puede estar falsificada, por lo que no se debe confiar en ella.

Cuando se reenvía una solicitud, el SDK crea un campo adicional en el segmento para indicar que se realizó esta acción. Si el segmento contiene el campo `x_forwarded_for` configurado en `true`, el IP del cliente se obtiene a partir del encabezado `X-Forwarded-For` en la solicitud HTTP.

El controlador de mensajes crea un segmento para cada solicitud entrante con un bloque `http` que contiene la siguiente información:

- Método HTTP: GET, POST, PUT, DELETE, etc.
- Dirección del cliente: la dirección IP del cliente que envió la solicitud.
- Código de respuesta: el código de respuesta HTTP para la solicitud finalizada.
- Intervalo: la hora de inicio (cuando se recibió la solicitud) y la hora de finalización (cuando se envió la respuesta).
- Agente del usuario: el `user-agent` de la solicitud.
- Longitud del contenido: la `content-length` de la respuesta.

Secciones

Seguimiento de solicitudes entrantes con Express

Para usar el middleware Express, inicie el cliente del SDK y utilice el middleware que devolvió la función `express.openSegment` antes de definir las rutas que desea usar.

Example app.js - Express

```
var app = express();

var AWSXRay = require('aws-xray-sdk');
app.use(AWSXRay.express.openSegment('MyApp'));

app.get('/', function (req, res) {
  res.render('index');
});

app.use(AWSXRay.express.closeSegment());
```

Después de definir las rutas, utilice el resultado de `express.closeSegment` tal como se muestra para poder solucionar los errores que el SDK de X-Ray para Node.js haya devuelto.

Seguimiento de solicitudes entrantes con Restify

Para utilizar el middleware Restify, inicialice el cliente del SDK y ejecute `enable`. Indique su servidor de Restify y el nombre de segmento.

Example app.js: restify

```
var AWSXRay = require('aws-xray-sdk');
var AWSXRayRestify = require('aws-xray-sdk-restify');

var restify = require('restify');
var server = restify.createServer();
AWSXRayRestify.enable(server, 'MyApp'));

server.get('/', function (req, res) {
  res.render('index');
});
```

Configuración de una estrategia de nomenclatura de segmentos

AWS X-Ray utiliza un nombre de servicio para identificar la aplicación y distinguirla del resto de aplicaciones, bases de datos, recursos externos APIs y otros AWS recursos que utiliza la aplicación. Cuando el SDK de X-Ray genera segmentos para las solicitudes entrantes, registra el nombre del servicio de la aplicación en el campo de nombre del segmento.

El SDK de X-Ray puede nombrar los segmentos utilizando el nombre de host en el encabezado de la solicitud HTTP. Sin embargo, este encabezado se puede falsificar, lo que podría provocar nodos inesperados en el mapa de servicio. Para evitar que el SDK nombre los segmentos de forma incorrecta debido a que las solicitudes tienen encabezados de host falsificados, debe especificar un nombre predeterminado para las solicitudes entrantes.

Si la aplicación atiende solicitudes de varios dominios, puede configurar el SDK para que utilice una estrategia de nomenclatura dinámica que refleje esto en los nombres de los segmentos. Una estrategia de nomenclatura dinámica permite al SDK usar el nombre de host para las solicitudes que coinciden con un patrón esperado y aplicar el nombre predeterminado a las solicitudes que no coincidan.

Por ejemplo, es posible que tenga una sola aplicación que atienda solicitudes a tres subdominios: `www.example.com`, `api.example.com` y `static.example.com`. Puede usar una estrategia de nomenclatura dinámica con el patrón `*.example.com` para identificar los segmentos de cada subdominio con un nombre diferente, lo que da como resultado tres nodos de servicio en el mapa de servicio. Si su aplicación recibe solicitudes con un nombre de host que no coincide con el patrón, verá un cuarto nodo en el mapa de servicio con el nombre alternativo que especifique.

Si desea utilizar el mismo nombre para todos los segmentos de solicitud, especifique el nombre de la aplicación cuando inicie el middleware, tal y como se muestra en las secciones anteriores.

> **Note**
>
> Puede anular el nombre de servicio predeterminado que ha definido en el código mediante la `AWS_XRAY_TRACING_NAME`variable de entorno???.

Una estrategia de nomenclatura dinámica define un patrón con el que deben coincidir los nombres de host y un nombre predeterminado que se utiliza si el nombre de host de la solicitud HTTP no coincide con el patrón. Para nombrar los segmentos dinámicamente, use `AWSXRay.middleware.enableDynamicNaming`.

Example app.js: nombres de segmentos dinámicos

Si el nombre de host de la solicitud coincide con el patrón `*.example.com`, utilice el nombre de host. De lo contrario, utilice `MyApp`.

```
var app = express();
```

```
var AWSXRay = require('aws-xray-sdk');
app.use(AWSXRay.express.openSegment('MyApp'));
AWSXRay.middleware.enableDynamicNaming('*.example.com');

app.get('/', function (req, res) {
  res.render('index');
});

app.use(AWSXRay.express.closeSegment());
```

Rastreo de llamadas al AWS SDK con el SDK de X-Ray para Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando la aplicación realiza llamadas Servicios de AWS para almacenar datos, escribir en una cola o enviar notificaciones, el SDK de X-Ray para Node.js rastrea las llamadas en sentido descendente en subsegmentos. El Servicios de AWS rastreo y los recursos a los que accede dentro de esos servicios (por ejemplo, un bucket de Amazon S3 o una cola de Amazon SQS) aparecen como nodos descendentes en el mapa de rastreo de la consola X-Ray.

Clientes del AWS SDK de instrumentos que se crean a través de la V2 o la AWS SDK para JavaScript V3.AWS SDK para JavaScript Cada versión AWS del SDK proporciona diferentes métodos para instrumentar AWS los clientes del SDK.

> **Note**
>
> Actualmente, el AWS X-Ray SDK para Node.js devuelve menos información de segmentos al instrumentar los clientes de la AWS SDK para JavaScript V3, en comparación con la instrumentación de los clientes de la V2. Por ejemplo, los subsegmentos que representan

llamadas a DynamoDB no devolverán el nombre de la tabla. Si necesita esta información de segmento en sus trazas, considere la posibilidad de utilizar la V2. AWS SDK para JavaScript

AWS SDK para JavaScript V2

Puede instrumentar todos los clientes AWS del SDK V2 empaquetando la sentencia `aws-sdk` require en una llamada a`AWSXRay.captureAWS`.

Example app.js: instrumentación con el SDK de AWS

```
const AWS = AWSXRay.captureAWS(require('aws-sdk'));
```

Para instrumentar a clientes individuales, incluye tu cliente del AWS SDK en una llamada a`AWSXRay.captureAWSClient`. Por ejemplo, para instrumentar un cliente de `AmazonDynamoDB`:

Example app.js - Instrumentación de clientes de DynamoDB

```
    const AWSXRay = require('aws-xray-sdk');
...
    const ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
```

> **Warning**
>
> No use `captureAWS` y `captureAWSClient` conjuntamente. Esto dará lugar a subsegmentos duplicados.

Si quieres usar TypeScriptECMAScriptmódulos (ESM) para cargar tu JavaScript código, usa el siguiente ejemplo para importar bibliotecas:

Example app.js: instrumentación AWS del SDK

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

Para instrumentar ESM AWS a todos los clientes, utilice el siguiente código:

Example app.js: instrumentación AWS del SDK

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
const XRAY_AWS = AWSXRay.captureAWS(AWS);
const ddb = new XRAY_AWS.DynamoDB();
```

Para todos los servicios, puede ver el nombre de la API a la que se llama en la consola de X-Ray. Para un subconjunto de servicios, el SDK de X-Ray agrega información al segmento para proporcionar una mayor granularidad en el mapa de servicio.

Por ejemplo, cuando realiza una llamada con un cliente instrumentado de DynamoDB, el SDK agrega el nombre de tabla al segmento para las llamadas que se dirigen a una tabla. En la consola, cada tabla aparece como nodo independiente en el mapa de servicio, con un nodo genérico de DynamoDB para las llamadas que no se dirigen a una tabla.

Example Subsegmento para una llamada a DynamoDB con el fin de guardar un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Cuando accede a recursos designados, las llamadas a los siguientes servicios crean nodos adicionales en el mapa de servicio. Las llamadas que no están dirigidas a recursos concretos crean un nodo genérico en el servicio.

- Amazon DynamoDB: nombre de tabla

- Amazon Simple Storage Service: nombre de bucket y de clave
- Amazon Simple Queue Service: nombre de cola

AWS SDK para JavaScript V3

La AWS SDK para JavaScript V3 es modular, por lo que su código solo carga los módulos que necesita. Por este motivo, no es posible instrumentar todos los clientes AWS del SDK, ya que la V3 no admite este método. `captureAWS`

Si quieres usar TypeScript ECMAScript Modules (ESM) para cargar tu JavaScript código, puedes usar el siguiente ejemplo para importar bibliotecas:

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

Instrumente cada cliente AWS del SDK mediante el `AWSXRay.captureAWSv3Client` método. Por ejemplo, para instrumentar un cliente de `AmazonDynamoDB`:

Example app.js: instrumentación de clientes de DynamoDB mediante la V3 del SDK para Javascript

```
    const AWSXRay = require('aws-xray-sdk');
    const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
...
    const ddb = AWSXRay.captureAWSv3Client(new DynamoDBClient({ region:
 "region" }));
```

Cuando se utiliza la AWS SDK para JavaScript V3, actualmente no se devuelven metadatos como el nombre de la tabla, el nombre del bucket y la clave o el nombre de la cola y, por lo tanto, el mapa de rastreo no contendrá nodos discretos para cada recurso nombrado, como ocurre cuando se instrumentan los clientes del AWS SDK mediante la V2. AWS SDK para JavaScript

Example Subsegmento para una llamada a DynamoDB para guardar un elemento, cuando se utiliza la V3 AWS SDK para JavaScript

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
```

```
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Rastreo de llamadas a servicios web HTTP posteriores utilizando el SDK de X-Ray para Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando tu aplicación realiza llamadas a microservicios o a HTTP públicos APIs, puedes usar el X-Ray SDK para el cliente Node.js para instrumentar esas llamadas y añadir la API al gráfico del servicio como un servicio descendente.

Pase su cliente `http` o `https` al método `captureHTTPs` del SDK de X-Ray para Node.js con el fin de rastrear llamadas salientes.

> **Note**
>
> Las llamadas que utilizan bibliotecas de solicitudes HTTP de terceros, como Axios o Superagent, son compatibles a través de la API `captureHTTPsGlobal()` y se seguirán rastreando cuando utilicen el módulo `http` nativo.

Example app.js: cliente HTTP

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Para habilitar el rastreo en todos los clientes HTTP, llame a `captureHTTPsGlobal` antes de cargar `http`.

Example app.js: cliente HTTP (global)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

Cuando se instrumenta una llamada a una API web posterior, el SDK de X-Ray para Node.js registra un subsegmento que contiene información acerca de la solicitud HTTP y la respuesta. X-Ray utiliza el subsegmento para generar un segmento inferido de la API remota.

Example Subsegmento para una llamada HTTP posterior

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
```

```
    }
  }
}
```

Example Segmento inferido para una llamada HTTP posterior

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

Rastreo de consultas SQL con el SDK de X-Ray para Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Instrumente las consultas de base de datos SQL incluyendo su cliente SQL en el método del cliente de SDK de X-Ray para Node.js correspondiente.

- PostgreSQL – `AWSXRay.capturePostgres()`

```
var AWSXRay = require('aws-xray-sdk');
var pg = AWSXRay.capturePostgres(require('pg'));
var client = new pg.Client();
```

- MySQL – `AWSXRay.captureMySQL()`

```
var AWSXRay = require('aws-xray-sdk');
var mysql = AWSXRay.captureMySQL(require('mysql'));
...
var connection = mysql.createConnection(config);
```

Cuando usa un cliente instrumentado para realizar consultas SQL, el SDK de X-Ray para Node.js registra información acerca de la conexión y consultas en un subsegmento.

Inclusión de datos adicionales en subsegmentos SQL

Puede agregar información adicional a los subsegmentos generados para consultas SQL, siempre que se asigne a un campo SQL con permiso. Por ejemplo, para registrar la cadena de consultas SQL saneada en un subsegmento, puede agregarla directamente al objeto SQL del subsegmento.

Example Asignar SQL al subsegmento

```
    const queryString = 'SELECT * FROM MyTable';
connection.query(queryString, ...);

// Retrieve the most recently created subsegment
const subs = AWSXRay.getSegment().subsegments;

if (subs & & subs.length > 0) {
  var sqlSub = subs[subs.length - 1];
  sqlSub.sql.sanitized_query = queryString;
}
```

Para obtener una lista completa de campos SQL con permiso, consulte Consultas SQL en la Guía para desarrolladores de AWS X-Ray .

Generación de subsegmentos personalizados con el SDK de X-Ray para Node.js

Los subsegmentos amplían el segmento de un rastro con detalles sobre el trabajo realizado para atender una solicitud. Cada vez que usted realiza una llamada con un cliente instrumentado, el SDK de X-Ray registra la información generada en un subsegmento. Puede crear subsegmentos adicionales para agrupar otros subsegmentos, medir el rendimiento de una sección de código o registrar anotaciones y metadatos.

Subsegmentos Express personalizados

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Para crear un subsegmento personalizado para una función que realiza llamadas a servicios posteriores, utilice la función `captureAsyncFunc`.

Example app.js: subsegmentos personalizados Express

```
var AWSXRay = require('aws-xray-sdk');

app.use(AWSXRay.express.openSegment('MyApp'));

app.get('/', function (req, res) {
  var host = 'api.example.com';

  AWSXRay.captureAsyncFunc('send', function(subsegment) {
    sendRequest(host, function() {
      console.log('rendering!');
      res.render('index');
      subsegment.close();
```

```
    });
  });
});

app.use(AWSXRay.express.closeSegment());

function sendRequest(host, cb) {
  var options = {
    host: host,
    path: '/',
  };

  var callback = function(response) {
    var str = '';

    response.on('data', function (chunk) {
      str += chunk;
    });

    response.on('end', function () {
      cb();
    });
  }

  http.request(options, callback).end();
};
```

En este ejemplo, la aplicación crea un subsegmento personalizado denominado `send` para realizar llamadas a la función `sendRequest`. `captureAsyncFunc` transfiere un subsegmento que debe cerrar dentro de la función de devolución de llamada cuando se completen las llamadas asíncronas que realiza.

Para las funciones síncronas, puede utilizar la función `captureFunc`, la cual cierra de forma automática el subsegmento en cuanto el bloque de funciones termina de ejecutarse.

Cuando crea un subsegmento dentro de un segmento o de otro subsegmento, el SDK de X-Ray para Node.js genera un ID para dicho subsegmento y registra la hora de inicio y la hora de finalización.

Example Subsegmentos con metadatos

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
```

```
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

Subsegmentos personalizados de Lambda

El SDK está configurado para crear automáticamente un segmento de fachada de marcador de posición cuando detecta que se está ejecutando en Lambda. Para crear un subsegmento básico, que creará un nodo `AWS::Lambda::Function` único en el mapa de rastros de X-Ray, llame y reasigne el segmento de fachada. Si crea manualmente un nuevo segmento con un nuevo ID (mientras comparte el ID de registro de seguimiento, el ID principal y la decisión de muestreo) podrá enviar un nuevo segmento.

Example app.js - subsegmentos personalizados manuales

```
const segment = AWSXRay.getSegment(); //returns the facade segment
const subsegment = segment.addNewSubsegment('subseg');
...
subsegment.close();
//the segment is closed by the SDK automatically
```

Adición de anotaciones y metadatos a los segmentos con el SDK de X-Ray para Node.js

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede registrar información adicional acerca de las solicitudes, el entorno o su aplicación con anotaciones y metadatos. Puede añadir anotaciones y metadatos a los segmentos que crea el SDK de X-Ray o a los subsegmentos personalizados que cree usted mismo.

Las anotaciones son pares de clave-valor con valores de cadena, numéricos o booleanos. Las anotaciones se indexan para su uso con expresiones de filtro. Utilice anotaciones para registrar los datos que desee utilizar para agrupar rastros en la consola o cuando llame a la API de `GetTraceSummaries`.

Los metadatos son pares de clave-valor con valores de cualquier tipo, por ejemplo objetos y listas, pero que no se indexan para utilizarlos con expresiones de filtro. Utilice los metadatos para registrar datos adicionales que desee almacenar en el rastro, pero que no necesite usar para hacer búsquedas.

Además de anotaciones y metadatos, también puede registrar cadenas de ID de usuario en los segmentos. IDs Los usuarios se registran en un campo independiente en los segmentos y se indexan para usarlos en la búsqueda.

Secciones

- Registro de anotaciones con el SDK de X-Ray para Node.js
- Registro de metadatos con el SDK de X-Ray para Node.js
- Grabación del usuario IDs con el SDK de X-Ray para Node.js

Registro de anotaciones con el SDK de X-Ray para Node.js

Utilice anotaciones para registrar información sobre segmentos o subsegmentos que desee indexar para las búsquedas.

Requisitos de anotación

- Claves: la clave de una anotación de X-Ray puede contener hasta 500 caracteres alfanuméricos. No se pueden usar espacios ni símbolos, excepto el punto (.)
- Valores: el valor de una anotación de X-Ray puede contener hasta 1000 caracteres Unicode.
- Número de anotaciones: se pueden utilizar hasta 50 anotaciones por rastro.

Para registrar anotaciones

1. Obtenga una referencia al segmento o subsegmento actual.

```
var AWSXRay = require('aws-xray-sdk');
...
var document = AWSXRay.getSegment();
```

2. Llame a `addAnnotation` con una clave de cadena y un valor booleano, numérico o de cadena.

```
document.addAnnotation("mykey", "my value");
```

   El siguiente ejemplo muestra cómo llamar a `putAnnotation` con una clave de cadena que incluye un punto y un valor booleano, numérico o de cadena.

```
document.putAnnotation("testkey.test", "my value");
```

El SDK registra las anotaciones como pares de clave-valor en un objeto `annotations` del documento de segmento. Si llama dos veces a `addAnnotation` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Para encontrar rastros que tengan anotaciones con valores específicos, utilice la palabra clave `annotation[`*`key`*`]` en una expresión de filtro.

Example app.js: anotaciones

```
var AWS = require('aws-sdk');
var AWSXRay = require('aws-xray-sdk');
var ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
...
app.post('/signup', function(req, res) {
    var item = {
        'email': {'S': req.body.email},
        'name': {'S': req.body.name},
        'preview': {'S': req.body.previewAccess},
        'theme': {'S': req.body.theme}
    };

    var seg = AWSXRay.getSegment();
    seg.addAnnotation('theme', req.body.theme);

    ddb.putItem({
      'TableName': ddbTable,
      'Item': item,
```

```
      'Expected': { email: { Exists: false } }
  }, function(err, data) {
...
```

Registro de metadatos con el SDK de X-Ray para Node.js

Utilice los metadatos para registrar información sobre segmentos o subsegmentos que no necesite indexar para las búsquedas. Los valores de metadatos pueden ser cadenas, números, booleanos o cualquier otro objeto que se pueda serializar en un objeto o matriz JSON.

Para registrar metadatos

1. Obtenga una referencia al segmento o subsegmento actual.

   ```
   var AWSXRay = require('aws-xray-sdk');
   ...
   var document = AWSXRay.getSegment();
   ```

2. Llame a `addMetadata` con una clave de cadena, un valor booleano, numérico, de cadena o de objeto y un espacio de nombres de cadena.

   ```
   document.addMetadata("my key", "my value", "my namespace");
   ```

   o

   Llame a `addMetadata` con solo una clave y un valor.

   ```
   document.addMetadata("my key", "my value");
   ```

Si no especifica un espacio de nombres, el SDK utiliza `default`. Si llama dos veces a `addMetadata` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Grabación del usuario IDs con el SDK de X-Ray para Node.js

Registre IDs el usuario en los segmentos de solicitud para identificar al usuario que envió la solicitud. Esta operación no es compatible con AWS Lambda las funciones porque los segmentos de los entornos Lambda son inmutables. La llamada `setUser` solo se puede efectuar con segmentos, no con subsegmentos.

Para registrar al usuario IDs

1. Obtenga una referencia al segmento o subsegmento actual.

```
var AWSXRay = require('aws-xray-sdk');
...
var document = AWSXRay.getSegment();
```

2. Llame a `setUser()` con un ID de cadena del usuario que envió la solicitud.

```
var user = 'john123';

AWSXRay.getSegment().setUser(user);
```

Puede llamar a `setUser` para registrar el ID de usuario en cuanto la aplicación rápida comience a procesar una solicitud. Si va a utilizar el segmento únicamente para establecer el ID de usuario, puede encadenar las llamadas en una sola línea.

Example app.js: ID de usuario

```
var AWS = require('aws-sdk');
var AWSXRay = require('aws-xray-sdk');
var uuidv4 = require('uuid/v4');
var ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
...
    app.post('/signup', function(req, res) {
    var userId = uuidv4();
    var item = {
        'userId': {'S': userId},
        'email': {'S': req.body.email},
        'name': {'S': req.body.name}
    };

    var seg = AWSXRay.getSegment().setUser(userId);

    ddb.putItem({
      'TableName': ddbTable,
      'Item': item,
      'Expected': { email: { Exists: false } }
  }, function(err, data) {
...
```

Para buscar rastros de un ID de usuario, utilice la palabra clave `user` en una expresión de filtro.

Trabajo con Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede instrumentar su aplicación Python de dos maneras para enviar rastros a X-Ray:

- AWS Distro for OpenTelemetry Python: una AWS distribución que proporciona un conjunto de bibliotecas de código abierto para enviar métricas y rastreos correlacionados a varias soluciones de AWS monitoreo, incluidas Amazon y Amazon OpenSearch Service CloudWatch AWS X-Ray, a través de AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK para Python: conjunto de bibliotecas para generar y enviar trazas a X-Ray mediante el daemon X-Ray.

Para obtener más información, consulte Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs.

AWS Distro para OpenTelemetry Python

Con AWS Distro para OpenTelemetry (ADOT) Python, puede instrumentar sus aplicaciones una vez y enviar métricas y rastros correlacionados a varias soluciones de supervisión de AWS, incluidas Amazon CloudWatch, AWS X-Ray y Amazon OpenSearch Service. El uso de X-Ray con ADOT requiere dos componentes: un SDK de OpenTelemetry habilitado para su uso con X-Ray y AWS Distro para OpenTelemetry Collector habilitado para su uso con X-Ray. ADOT Python admite la instrumentación automática, lo que permite a la aplicación del usuario enviar rastros sin cambios de código.

Para empezar, consulte la documentación de AWS Distro para OpenTelemetry Python.

Para obtener más información sobre el uso de AWS Distro para OpenTelemetry con AWS X-Ray y otros Servicios de AWS, consulte AWS Distro para OpenTelemetry o la documentación de AWS Distro para OpenTelemetry.

Para obtener información adicional sobre los lenguajes compatibles y el uso, consulte AWS Observability en GitHub.

AWS X-Ray SDK para Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Python es una biblioteca para las aplicaciones web de Python que proporciona clases y métodos para generar y enviar datos de rastreo al daemon de X-Ray. Los datos de rastreo incluyen información sobre las solicitudes HTTP entrantes atendidas por la aplicación y las llamadas que la aplicación realiza a los servicios descendentes mediante el AWS SDK, los clientes HTTP o un conector de base de datos SQL. También puede crear segmentos de forma manual y agregar información de depuración en anotaciones y metadatos.

Puede descargar el SDK con `pip`.

```
$ pip install aws-xray-sdk
```

> **Note**
>
> El SDK de X-Ray para Python es un proyecto de código abierto. Puedes seguir el proyecto y enviar las incidencias y solicitudes de cambios en GitHub: github. com/aws/aws- xray-sdk-python

Si usa Django o Flask, empiece agregando el middleware del SDK a la aplicación para rastrear las solicitudes entrantes. El middleware crea un segmento para cada solicitud rastreada y lo completa cuando se envía la respuesta. Mientras el segmento está abierto, puede utilizar los métodos del cliente del SDK para añadir información al segmento y crear subsegmentos para rastrear llamadas posteriores. El SDK también registra automáticamente las excepciones que produce su aplicación mientras el segmento está abierto. Para otras aplicaciones, puede crear segmentos manualmente.

En el caso de las funciones de Lambda llamadas por una aplicación o un servicio instrumentados, Lambda lee el encabezado de rastreo y rastrea automáticamente las solicitudes muestreadas. Para otras funciones, puede configurar Lambda con el fin de muestrear y rastrear las solicitudes entrantes. En cualquier caso, Lambda crea el segmento y se lo proporciona al SDK de X-Ray.

> **Note**
>
> En Lambda, el SDK de X-Ray es opcional. Si no lo usa en su función, el mapa de servicio seguirá incluyendo un nodo para el servicio de Lambda y uno para cada función de Lambda. Al añadir el SDK, puede instrumentar el código de función para añadir subsegmentos al segmento de función registrado por Lambda. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

Consulte Entorno de trabajo para ver un ejemplo de una función de Python instrumentada en Lambda.

A continuación, utilice el SDK de X-Ray para Python con el fin de instrumentar llamadas posteriores aplicando parches a las bibliotecas de su aplicación. El SDK admite las siguientes bibliotecas.

Bibliotecas compatibles

- `botocore`, `boto3` — AWS SDK para Python (Boto) Clientes de instrumentos.
- `pynamodb`: instrumente la versión de PynamoDB del cliente de Amazon DynamoDB.
- `aiobotocore`, `aioboto3`: instrumente las versiones integradas en asyncio de los clientes del SDK para Python.
- `requests`, `aiohttp`: instrumente clientes HTTP de alto nivel.
- `httplib`, `http.client`: instrumente los clientes HTTP de bajo nivel y las bibliotecas de más alto nivel que los utilizan.
- `sqlite3`— SQLite Clientes de instrumentos.

- `mysql-connector-python`: instrumente los clientes de MySQL.
- `pg8000`: instrumente la interfaz PostgreSQL Pure-Python.
- `psycopg2`: instrumente el adaptador de base de datos PostgreSQL.
- `pymongo`: instrumente clientes de MongoDB.
- `pymysql`— Clientes basados en Instrument PyMy SQL y MariaDB.

Cada vez que la aplicación realiza llamadas a AWS una base de datos SQL u otros servicios HTTP, el SDK registra la información sobre la llamada en un subsegmento. Servicios de AWS y los recursos a los que accedes desde los servicios aparecen como nodos descendentes en el mapa de rastreo para ayudarte a identificar los errores y los problemas de limitación en las conexiones individuales.

En cuanto empiece a utilizar el SDK, personalice su comportamiento configurando la grabadora y el controlador y el middleware. Puede añadir complementos para registrar los datos sobre los recursos informáticos que ejecutan su aplicación, personalizar el comportamiento de muestreo mediante la definición de reglas de muestreo y definir el nivel de log para ver más o menos información del SDK en los logs de las aplicaciones.

Registre información adicional acerca de las solicitudes y el trabajo que la aplicación realiza en anotaciones y metadatos. Las anotaciones son pares sencillos de clave-valor que se indexan para su uso con expresiones de filtro para poder buscar rastros que contengan datos específicos. Las entradas de metadatos son menos restrictivas y pueden registrar objetos y matrices completos, es decir, todo lo que se pueda serializar en JSON.

> **Anotaciones y metadatos**
>
> Las anotaciones y los metadatos son texto arbitrario que se agrega a los segmentos con el SDK de X-Ray. Las anotaciones se indexan para su uso con expresiones de filtro. Los metadatos no se indexan pero se pueden ver en el segmento sin procesar con la consola o la API de X-Ray. Cualquier persona a la que conceda acceso de lectura a X-Ray puede ver estos datos.

Cuando tenga muchos clientes instrumentados en su código, un único segmento de solicitud puede contener un gran número de subsegmentos, uno para cada llamada realizada con un cliente instrumentado. Puede organizar y agrupar los subsegmentos incluyendo las llamadas del cliente en subsegmentos personalizados. Puede crear un subsegmento personalizado para toda una

función o cualquier sección de código. A continuación, puede registrar metadatos y anotaciones en el subsegmento en lugar de escribir todo en el segmento principal.

Para acceder a la documentación de referencia de las clases y los métodos del SDK, consulte la Referencia de la API del SDK de AWS X-Ray para Python.

Requisitos

El SDK de X-Ray para Python es compatible con los siguientes lenguajes y versiones de biblioteca.

- Python: 2.7, 3.4 y posteriores
- Django: 1.10 y posteriores
- Flask: 0.10 y posteriores
- aiohttp: 2.3.0 y posteriores
- AWS SDK para Python (Boto): 1.4.0 y posteriores
- botocore: 1.5.0 y posteriores
- enum: 0.4.7 y posteriores, para Python 3.4.0 y anteriores
- jsonpickle: 1.0.0 y posteriores
- setuptools: 40.6.3 y posteriores
- wrapt: 1.11.0 y posteriores

Administración de dependencias

El SDK de X-Ray para Python está disponible en `pip`.

- Paquete: `aws-xray-sdk`

Añada el SDK como una dependencia en su archivo `requirements.txt`.

Example requirements.txt

```
aws-xray-sdk==2.4.2
boto3==1.4.4
botocore==1.5.55
Django==1.11.3
```

Si utiliza Elastic Beanstalk para implementar su aplicación, Elastic Beanstalk instala todos los paquetes en `requirements.txt` de forma automática.

Configuración del SDK de X-Ray para Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Python tiene una clase denominada `xray_recorder` que proporciona la grabadora global. Puede configurar la grabadora global para que personalice el middleware que crea los segmentos para las llamadas HTTP entrantes.

Secciones

- Complementos del servicio
- Reglas de muestreo
- Registro
- Configuración de la grabadora en código
- Configuración de la grabadora con Django
- Variables de entorno

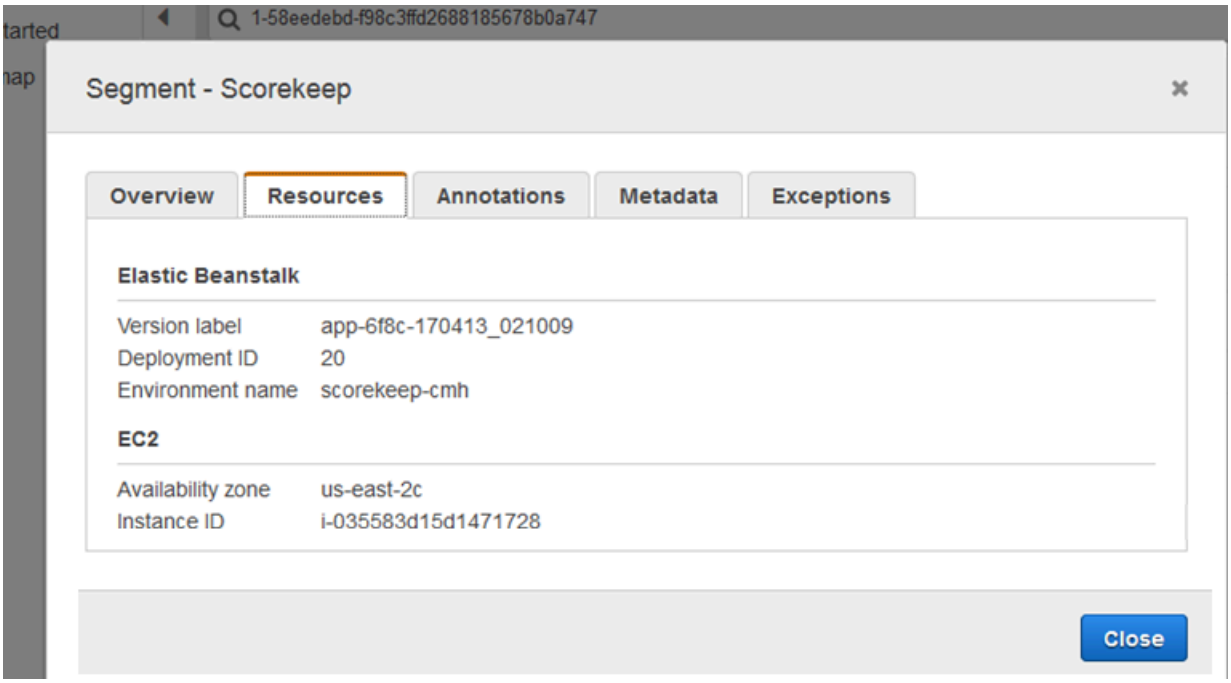
Complementos del servicio

Utilice `plugins` para registrar información sobre el servicio que aloja la aplicación.

Plugins

- Amazon EC2 : `EC2Plugin` agrega el ID de la instancia, la zona de disponibilidad y el grupo de CloudWatch registros.

- Elastic Beanstalk: `ElasticBeanstalkPlugin` añade el nombre de entorno, la etiqueta de versión y el ID de implementación.
- Amazon ECS: `ECSPlugin` agrega el ID de contenedor.



Para utilizar un complemento, llame a `configure` en el `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

xray_recorder.configure(service='My app')
plugins = ('ElasticBeanstalkPlugin', 'EC2Plugin')
xray_recorder.configure(plugins=plugins)
patch_all()
```

> **Note**
>
> Dado que se transfieren `plugins` como tupla, asegúrese de incluir una `,` después al especificar un único complemento. Por ejemplo, `plugins = ('EC2Plugin',)`

También puede utilizar las variables de entorno, que tienen prioridad frente a los valores establecidos en código, para configurar la grabadora.

Configure los complementos antes de aplicar parches en las bibliotecas para registrar las llamadas posteriores.

El SDK también usa la configuración del complemento para establecer el campo `origin` en el segmento. Esto indica el tipo de AWS recurso que ejecuta la aplicación. Cuando utilizas varios complementos, el SDK utiliza el siguiente orden de resolución para determinar el origen: ElasticBeanstalk > EKS > ECS > EC2.

Reglas de muestreo

El SDK utiliza las reglas de muestreo que define el usuario en la consola de X-Ray para determinar qué solicitudes registrar. La regla predeterminada rastrea la primera solicitud cada segundo y el 5 % de las solicitudes adicionales de todos los servicios que envían rastros a X-Ray. Cree reglas adicionales en la consola de X-Ray para personalizar la cantidad de datos registrados para cada una de sus aplicaciones.

El SDK aplica las reglas personalizadas en el orden en que se definen. Si una solicitud coincide con varias reglas personalizadas, el SDK solo aplica la primera regla.

> **Note**
>
> Si el SDK no puede comunicarse con X-Ray para obtener las reglas de muestreo, vuelve a una regla local predeterminada de la primera solicitud cada segundo y del 5 % de las solicitudes adicionales por host. Esto puede ocurrir si el host no tiene permiso para llamar al muestreo APIs o no puede conectarse al daemon X-Ray, que actúa como proxy TCP para las llamadas a la API realizadas por el SDK.

También puede configurar el SDK para que cargue las reglas de muestreo desde un documento JSON. El SDK puede usar las reglas locales como respaldo para los casos en que el muestreo de X-Ray no esté disponible, o puede usar las reglas locales exclusivamente.

Example sampling-rules.json

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
```

```
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

En este ejemplo se define una regla personalizada y una regla predeterminada. La regla personalizada aplica un porcentaje de muestreo del 5 % sin un número mínimo de solicitudes de rastreo para las rutas incluidas bajo `/api/move/`. La regla predeterminada rastrea la primera solicitud cada segundo y el 10 % de las solicitudes adicionales.

La desventaja de definir las reglas localmente es que el objetivo establecido lo aplica cada instancia de la grabadora de forma independiente, en lugar de ser administrado por el servicio de X-Ray. A medida que se implementan más hosts, el porcentaje establecido se multiplica, lo que dificulta el control de la cantidad de datos registrados.

Sí AWS Lambda, no puedes modificar la frecuencia de muestreo. Si un servicio instrumentado llama a su función, Lambda registrará las llamadas que generaron solicitudes muestreadas por ese servicio. Si el rastreo activo está activado y no hay ningún encabezado de rastreo, Lambda toma la decisión de muestreo.

Para configurar las reglas de muestreo de respaldo`xray_recorder.configure`, llame, como se muestra en el siguiente ejemplo, a un diccionario de reglas o a la ruta absoluta a un archivo JSON que contiene reglas de muestreo. *`rules`*

```
xray_recorder.configure(sampling_rules=rules)
```

Para utilizar solo reglas locales, configure la grabadora con un `LocalSampler`.

```
from aws_xray_sdk.core.sampling.local.sampler import LocalSampler
xray_recorder.configure(sampler=LocalSampler())
```

También puede configurar la grabadora global para deshabilitar el muestreo e instrumentar todas las solicitudes entrantes.

Example main.py: deshabilite el muestreo

```
xray_recorder.configure(sampling=False)
```

Registro

El SDK usa el módulo integrado `logging` de Python con un nivel de registro predeterminado `WARNING`. Obtenga una referencia al registrador para la clase `aws_xray_sdk` y llame `setLevel` en la misma para configurar el nivel de log diferente para la biblioteca y el resto de la aplicación.

Example app.py registro

```
logging.basicConfig(level='WARNING')
logging.getLogger('aws_xray_sdk').setLevel(logging.ERROR)
```

Utilice registros de depuración para identificar problemas como la presencia de subsegmentos sin cerrar al generar subsegmentos de forma manual.

Configuración de la grabadora en código

Hay disponibles ajustes adicionales a partir del método `configure` en `xray_recorder`.

- `context_missing`: establezca esta opción en `LOG_ERROR` para evitar que se produzcan excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.
- `daemon_address`: establezca el host y el puerto del oyente del daemon de X-Ray.
- `service`: establezca un nombre de servicio que el SDK utiliza para los segmentos.
- `plugins`— Registra información sobre los AWS recursos de tu aplicación.
- `sampling`: establezca esta opción en `False` para deshabilitar el muestreo.
- `sampling_rules`: establezca la ruta del archivo JSON que contiene sus reglas de muestreo.

Example main.py: deshabilite excepciones de falta de contexto

```
from aws_xray_sdk.core import xray_recorder

xray_recorder.configure(context_missing='LOG_ERROR')
```

Configuración de la grabadora con Django

Si utiliza el marco de Django, puede utilizar el archivo `settings.py` de Django para configurar opciones en la grabadora global.

- `AUTO_INSTRUMENT` (solo Django): registra subsegmentos para operaciones de representación de plantilla y base de datos integrada.
- `AWS_XRAY_CONTEXT_MISSING`: establezca esta opción en `LOG_ERROR` para evitar que se produzcan excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.
- `AWS_XRAY_DAEMON_ADDRESS`: establezca el host y el puerto del oyente del daemon de X-Ray.
- `AWS_XRAY_TRACING_NAME`: establezca un nombre de servicio que el SDK utiliza para los segmentos.
- `PLUGINS`— Registre la información sobre AWS los recursos de su aplicación.
- `SAMPLING`: establezca esta opción en `False` para deshabilitar el muestreo.
- `SAMPLING_RULES`: establezca la ruta del archivo JSON que contiene sus reglas de muestreo.

Para habilitar la configuración de la grabadora en `settings.py`, añada el middleware de Django a la lista de aplicaciones instaladas.

Example settings.py: aplicaciones instaladas

```
INSTALLED_APPS = [
    ...
    'django.contrib.sessions',
    'aws_xray_sdk.ext.django',
]
```

Configure los ajustes disponibles en un diccionario denominado `XRAY_RECORDER`.

Example settings.py: aplicaciones instaladas

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_CONTEXT_MISSING': 'LOG_ERROR',
    'AWS_XRAY_DAEMON_ADDRESS': '127.0.0.1:5000',
    'AWS_XRAY_TRACING_NAME': 'My application',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin', 'ECSPlugin'),
```

```
    'SAMPLING': False,
}
```

Variables de entorno

Puede usar variables de entorno con el fin de configurar el SDK de X-Ray para Python. El SDK admite las siguientes variables:

- `AWS_XRAY_TRACING_NAME`: establezca un nombre de servicio que el SDK utiliza para los segmentos. Anula el nombre de servicio que se establece mediante programación.
- `AWS_XRAY_SDK_ENABLED`: cuando se establece en `false`, deshabilita el SDK. De forma predeterminada, el SDK está habilitado a menos que la variable de entorno esté establecida en false.
  - Cuando está deshabilitado, el grabador global genera automáticamente segmentos y subsegmentos ficticios que no se envían al demonio y se deshabilita la aplicación automática de parches. El middleware se registra como un contenedor en el grabador local. Todos los segmentos y subsegmentos que se generan a través del middleware también se vuelven ficticios.
  - Establezca el valor de `AWS_XRAY_SDK_ENABLED` a través de la variable de entorno o interactuando directamente con el objeto `global_sdk_config` de la biblioteca `aws_xray_sdk`. La configuración de la variable de entorno invalida estas interacciones.
- `AWS_XRAY_DAEMON_ADDRESS`: establezca el host y el puerto del oyente del daemon de X-Ray. De forma predeterminada, el SDK utiliza `127.0.0.1:2000` tanto para los datos de rastro (UDP) como para el muestreo (TCP). Use esta variable si ha configurado el daemon para que escuche en un puerto diferente o si se ejecuta en un host diferente.

  Formato

  - El mismo puerto: *`address`*`:`*`port`*
  - Puertos diferentes: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_CONTEXT_MISSING`: establezca esta opción en `RUNTIME_ERROR` para generar excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.

  Valores válidos

  - `RUNTIME_ERROR`: lance una excepción de tiempo de ejecución.
  - `LOG_ERROR`: registre un error y continúe (predeterminado).

- `IGNORE_ERROR`: ignore el error y continúe.

  Se pueden producir errores relativos a segmentos o subsegmentos inexistentes al intentar usar un cliente instrumentado en el código de inicio que se ejecuta cuando no hay ninguna solicitud abierta o en el código que inicia un nuevo subproceso.

Las variables de entorno anulan los valores establecidos en el código.

Rastreo de las solicitudes entrantes con el SDK de X-Ray para middleware de Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando añade el middleware a su aplicación y configura un nombre de segmento, el SDK de X-Ray para Python crea un segmento para cada solicitud muestreada. Este segmento incluye el momento, el método y la disposición de la solicitud HTTP. La instrumentación adicional crea subsegmentos en este segmento.

El SDK de X-Ray para Python admite el siguiente middleware para instrumentar solicitudes HTTP entrantes:

- Django
- Flask
- Bottle

> **Note**
>
> Para AWS Lambda las funciones, Lambda crea un segmento para cada solicitud muestreada. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

Consulte Entorno de trabajo para ver un ejemplo de una función de Python instrumentada en Lambda.

Para scripts o aplicaciones Python en otros marcos de trabajo, puede crear los segmentos manualmente.

Cada segmento tiene un nombre que identifica la aplicación en el mapa de servicio. El nombre del segmento se puede asignar de forma estática o se puede configurar el SDK para que le asigne un nombre dinámico en función del encabezado del host de la solicitud entrante. La nomenclatura dinámica permite agrupar los rastros en función del nombre de dominio de la solicitud y aplicar un nombre predeterminado si el nombre no coincide con el patrón esperado (por ejemplo, si el encabezado del host está falsificado).

> **Solicitudes reenviadas**
>
> Si un equilibrador de carga u otro intermediario reenvía una solicitud a la aplicación, X-Ray toma la IP de cliente del encabezado `X-Forwarded-For` de la solicitud en lugar de tomar la IP de origen del paquete IP. La IP de cliente que se graba para una solicitud reenviada puede estar falsificada, por lo que no se debe confiar en ella.

Cuando se reenvía una solicitud, el SDK crea un campo adicional en el segmento para indicar que se realizó esta acción. Si el segmento contiene el campo `x_forwarded_for` configurado en `true`, el IP del cliente se obtiene a partir del encabezado `X-Forwarded-For` en la solicitud HTTP.

El middleware crea un segmento para cada solicitud entrante con un bloque `http` que contiene la siguiente información:

- Método HTTP: GET, POST, PUT, DELETE, etc.
- Dirección del cliente: la dirección IP del cliente que envió la solicitud.
- Código de respuesta: el código de respuesta HTTP para la solicitud finalizada.

- Intervalo: la hora de inicio (cuando se recibió la solicitud) y la hora de finalización (cuando se envió la respuesta).
- Agente del usuario: el `user-agent` de la solicitud.
- Longitud del contenido: la `content-length` de la respuesta.

Secciones

Agregar el middleware a su aplicación (Django)

Añada el middleware a la lista `MIDDLEWARE` en su archivo `settings.py`. El middleware de X-Ray debe ser la primera línea de su archivo `settings.py` para garantizar que se registren las solicitudes que fallan en otros middleware.

Example settings.py: SDK de X-Ray para middleware de Python

```
MIDDLEWARE = [
    'aws_xray_sdk.ext.django.middleware.XRayMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware'
]
```

Añada la aplicación Django del SDK de X-Ray a la lista `INSTALLED_APPS` de su archivo `settings.py`. Eso permitirá configurar la grabadora de X-Ray durante el inicio de la aplicación.

Example settings.py: aplicación Django del SDK de X-Ray para Python

```
INSTALLED_APPS = [
```

```
    'aws_xray_sdk.ext.django',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Configure un nombre de segmento en su `settings.py` archivo.

Example settings.py: nombre de segmento

```
XRAY_RECORDER = {
    'AWS_XRAY_TRACING_NAME': 'My application',
    'PLUGINS': ('EC2Plugin',),
}
```

Esto indica a la grabadora de X-Ray que rastree solicitudes servidas por la aplicación Django con el porcentaje de muestreo predeterminado. Puede configurar la grabadora con el archivo de configuración de Django para aplicar reglas de muestreo personalizadas o cambiar otros ajustes.

> **Note**
>
> Dado que se transfieren `plugins` como tupla, asegúrese de incluir una `,` después al especificar un único complemento. Por ejemplo, `plugins = ('EC2Plugin',)`

Agregar el middleware a su aplicación (flask)

Para instrumentar su aplicación Flask, en primer lugar configure un nombre de segmento en el `xray_recorder`. A continuación, utilice la función `XRayMiddleware` para aplicar un parche a su aplicación Flask en código.

Example app.py

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.ext.flask.middleware import XRayMiddleware

app = Flask(__name__)
```

```
xray_recorder.configure(service='My application')
XRayMiddleware(app, xray_recorder)
```

Esto indica a la grabadora de X-Ray que rastree solicitudes servidas por la aplicación Flask con el porcentaje de muestreo predeterminado. Puede configurar la grabadora en código para aplicar reglas de muestreo personalizadas o cambiar otros ajustes.

Agregar el middleware a su aplicación (Bottle)

Para instrumentar su aplicación Bottle, en primer lugar configure un nombre de segmento en el `xray_recorder`. A continuación, utilice la función `XRayMiddleware` para aplicar un parche a su aplicación Bottle en código.

Example app.py

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.ext.bottle.middleware import XRayMiddleware

app = Bottle()

xray_recorder.configure(service='fallback_name', dynamic_naming='My application')
app.install(XRayMiddleware(xray_recorder))
```

Esto indica a la grabadora de X-Ray que rastree solicitudes servidas por la aplicación Bottle con el porcentaje de muestreo predeterminado. Puede configurar la grabadora en código para aplicar reglas de muestreo personalizadas o cambiar otros ajustes.

Instrumentación manual del código de Python

Si no utiliza Django o Flask, puede crear segmentos manualmente. Puede crear un segmento para cada solicitud entrante o crear segmentos en torno a clientes HTTP o AWS SDK parcheados para proporcionar un contexto que permita a la grabadora añadir subsegmentos.

Example main.py: instrumentación manual

```
from aws_xray_sdk.core import xray_recorder

# Start a segment
segment = xray_recorder.begin_segment('segment_name')
# Start a subsegment
subsegment = xray_recorder.begin_subsegment('subsegment_name')
```

```
# Add metadata and annotations
segment.put_metadata('key', dict, 'namespace')
subsegment.put_annotation('key', 'value')

# Close the subsegment and segment
xray_recorder.end_subsegment()
xray_recorder.end_segment()
```

Configuración de una estrategia de nomenclatura de segmentos

AWS X-Ray utiliza un nombre de servicio para identificar la aplicación y distinguirla del resto de aplicaciones, bases de datos, recursos externos APIs y otros AWS recursos que utiliza la aplicación. Cuando el SDK de X-Ray genera segmentos para las solicitudes entrantes, registra el nombre del servicio de la aplicación en el campo de nombre del segmento.

El SDK de X-Ray puede nombrar los segmentos utilizando el nombre de host en el encabezado de la solicitud HTTP. Sin embargo, este encabezado se puede falsificar, lo que podría provocar nodos inesperados en el mapa de servicio. Para evitar que el SDK nombre los segmentos de forma incorrecta debido a que las solicitudes tienen encabezados de host falsificados, debe especificar un nombre predeterminado para las solicitudes entrantes.

Si la aplicación atiende solicitudes de varios dominios, puede configurar el SDK para que utilice una estrategia de nomenclatura dinámica que refleje esto en los nombres de los segmentos. Una estrategia de nomenclatura dinámica permite al SDK usar el nombre de host para las solicitudes que coinciden con un patrón esperado y aplicar el nombre predeterminado a las solicitudes que no coincidan.

Por ejemplo, es posible que tenga una sola aplicación que atienda solicitudes a tres subdominios: `www.example.com`, `api.example.com` y `static.example.com`. Puede usar una estrategia de nomenclatura dinámica con el patrón `*.example.com` para identificar los segmentos de cada subdominio con un nombre diferente, lo que da como resultado tres nodos de servicio en el mapa de servicio. Si su aplicación recibe solicitudes con un nombre de host que no coincide con el patrón, verá un cuarto nodo en el mapa de servicio con el nombre alternativo que especifique.

Si desea utilizar el mismo nombre para todos los segmentos de solicitud, especifique el nombre de la aplicación cuando configure la grabadora, como se muestra en las secciones anteriores.

Una estrategia de nomenclatura dinámica define un patrón con el que deben coincidir los nombres de host y un nombre predeterminado que se utiliza si el nombre de host de la solicitud HTTP

no coincide con el patrón. Para nombrar segmentos de forma dinámica en Django, añada la configuración DYNAMIC_NAMING a su archivo settings.py.

Example settings.py: nomenclatura dinámica

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_TRACING_NAME': 'My application',
    'DYNAMIC_NAMING': '*.example.com',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin')
}
```

Puede utilizar "*" en el patrón para buscar coincidencia con cualquier cadena o "?" para buscar coincidencias con cualquier carácter único. Para Flask, configure la grabadora en código.

Example main.py: nombre de segmento

```
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='My application')
xray_recorder.configure(dynamic_naming='*.example.com')
```

> **Note**
>
> Puede anular el nombre de servicio predeterminado que ha definido en el código mediante la AWS_XRAY_TRACING_NAMEvariable de entorno???.

Aplicación de parches a bibliotecas para instrumentar llamadas posteriores

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Para instrumentar llamadas posteriores, utilice el SDK de X-Ray para Python para aplicar parches a las bibliotecas que utiliza la aplicación. El SDK de X-Ray para Python puede aplicar parches a las siguientes bibliotecas.

Bibliotecas compatibles

- `botocore`, `boto3` — Clientes de instrumentos. AWS SDK para Python (Boto)
- `pynamodb`: instrumente la versión de PynamoDB del cliente de Amazon DynamoDB.
- `aiobotocore`, `aioboto3`: instrumente las versiones integradas en asyncio de los clientes del SDK para Python.
- `requests`, `aiohttp`: instrumente clientes HTTP de alto nivel.
- `httplib`, `http.client`: instrumente los clientes HTTP de bajo nivel y las bibliotecas de más alto nivel que los utilizan.
- `sqlite3`— SQLite Clientes de instrumentos.
- `mysql-connector-python`: instrumente los clientes de MySQL.
- `pg8000`: instrumente la interfaz PostgreSQL Pure-Python.
- `psycopg2`: instrumente el adaptador de base de datos PostgreSQL.
- `pymongo`: instrumente clientes de MongoDB.
- `pymysql`— Clientes basados en Instrument PyMy SQL y MariaDB.

Al utilizar una biblioteca con parches, el SDK de X-Ray para Python crea un subsegmento para la llamada y registra información desde la solicitud y la respuesta. Debe haber un segmento disponible para que el SDK cree el subsegmento, ya sea desde el middleware del SDK o desde AWS Lambda.

> **Note**
>
> Si utilizas SQLAlchemy ORM, puedes instrumentar tus consultas SQL importando la versión del SDK de las clases SQLAlchemy de sesión y consulta. Consulta Cómo usar SQLAlchemy ORM para obtener instrucciones.

Para aplicar parches a todas las bibliotecas disponibles, utilice la función `patch_all` en `aws_xray_sdk.core`. Es posible que algunas bibliotecas, como `httplib` y `urllib`, necesiten habilitar la aplicación de parches dobles llamando a `patch_all(double_patch=True)`.

Example main.py: aplique parches a todas las bibliotecas compatibles

```
import boto3
import botocore
import requests
import sqlite3

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

patch_all()
```

Para aplicar parches a una sola biblioteca, llame a `patch` con una tupla del nombre de la biblioteca. Para ello, deberá proporcionar una sola lista de elementos.

Example main.py: aplique parches a bibliotecas específicas

```
import boto3
import botocore
import requests
import mysql-connector-python

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch

libraries = ('botocore')
patch(libraries)
```

> **Note**
>
> En algunos casos, la clave que se utiliza aplicar parches a una biblioteca no coincide con el nombre de la biblioteca. Algunas claves sirven como alias para una o varias bibliotecas.
>
> Alias de las bibliotecas
>
> - `httplib`: `httplib` y `http.client`
> - `mysql` – `mysql-connector-python`

Seguimiento del contexto para el funcionamiento asíncrono

Para las bibliotecas integradas de `asyncio` o para crear subsegmentos para funciones asíncronas, también debe configurar el SDK de X-Ray para Python con un contexto asíncrono. Importe la clase `AsyncContext` y pase una instancia de ella a la grabadora de X-Ray.

> **Note**
>
> Las bibliotecas compatibles con el marco de trabajo web, como AIOHTTP, no se gestionan a través del módulo `aws_xray_sdk.core.patcher`. No aparecerán en el catálogo de bibliotecas admitidas de`patcher`.

Example main.py: aplique parches a aioboto3

```
import asyncio
import aioboto3
import requests

from aws_xray_sdk.core.async_context import AsyncContext
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='my_service', context=AsyncContext())
from aws_xray_sdk.core import patch

libraries = (['aioboto3'])
patch(libraries)
```

Rastreo de llamadas al AWS SDK con el SDK de X-Ray para Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando la aplicación realiza llamadas Servicios de AWS para almacenar datos, escribir en una cola o enviar notificaciones, el SDK de X-Ray para Python rastrea las llamadas en sentido descendente en subsegmentos. El rastreo Servicios de AWS y los recursos a los que accede dentro de esos servicios (por ejemplo, un bucket de Amazon S3 o una cola de Amazon SQS) aparecen como nodos descendentes en el mapa de rastreo de la consola X-Ray.

El SDK de X-Ray para Python instrumenta automáticamente todos los clientes AWS del SDK cuando se aplica un parche a la `botocore` biblioteca. No puede instrumentar clientes individuales.

Para todos los servicios, puede ver el nombre de la API a la que se llama en la consola de X-Ray. Para un subconjunto de servicios, el SDK de X-Ray agrega información al segmento para proporcionar una mayor granularidad en el mapa de servicio.

Por ejemplo, cuando realiza una llamada con un cliente instrumentado de DynamoDB, el SDK agrega el nombre de tabla al segmento para las llamadas que se dirigen a una tabla. En la consola, cada tabla aparece como nodo independiente en el mapa de servicio, con un nodo genérico de DynamoDB para las llamadas que no se dirigen a una tabla.

Example Subsegmento para una llamada a DynamoDB con el fin de guardar un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Cuando accede a recursos designados, las llamadas a los siguientes servicios crean nodos adicionales en el mapa de servicio. Las llamadas que no están dirigidas a recursos concretos crean un nodo genérico en el servicio.

- Amazon DynamoDB: nombre de tabla
- Amazon Simple Storage Service: nombre de bucket y de clave
- Amazon Simple Queue Service: nombre de cola

Rastreo de llamadas a servicios web HTTP posteriores utilizando el SDK de X-Ray para Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando tu aplicación realiza llamadas a microservicios o HTTP públicos APIs, puedes usar el SDK de X-Ray para Python para instrumentar esas llamadas y añadir la API al gráfico del servicio como un servicio descendente.

Para instrumentar clientes HTTP, debe aplicar parches a la biblioteca que se utiliza para realizar llamadas salientes. Si utiliza `requests` o el cliente HTTP integrado de Python, eso es lo único que tiene que hacer. Para `aiohttp`, también debe configurar la grabadora con un contexto asíncrono.

Si utiliza la API del cliente de `aiohttp` 3, es posible que también deba configurar la sesión de `ClientSession` con una instancia de la configuración de seguimiento proporcionada en el SDK.

Example API del cliente 3 de **`aiohttp`**

```
from aws_xray_sdk.ext.aiohttp.client import aws_xray_trace_config
```

```
async def foo():
    trace_config = aws_xray_trace_config()
    async with ClientSession(loop=loop, trace_configs=[trace_config]) as session:
        async with session.get(url) as resp
            await resp.read()
```

Cuando se instrumenta una llamada a una API web posterior, el SDK de X-Ray para Python registra un subsegmento que contiene información acerca de la solicitud HTTP y la respuesta. X-Ray utiliza el subsegmento para generar un segmento inferido de la API remota.

Example Subsegmento para una llamada HTTP posterior

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example Segmento inferido para una llamada HTTP posterior

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
```

```
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

Generación de subsegmentos personalizados con el SDK de X-Ray para Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Los subsegmentos amplían el segmento de un rastro con detalles sobre el trabajo realizado para atender una solicitud. Cada vez que usted realiza una llamada con un cliente instrumentado, el SDK de X-Ray registra la información generada en un subsegmento. Puede crear subsegmentos adicionales para agrupar otros subsegmentos, medir el rendimiento de una sección de código o registrar anotaciones y metadatos.

Para administrar los subsegmentos, utilice los métodos `begin_subsegment` y `end_subsegment`.

Example main.py: subsegmento personalizado

```
from aws_xray_sdk.core import xray_recorder

subsegment = xray_recorder.begin_subsegment('annotations')
subsegment.put_annotation('id', 12345)
xray_recorder.end_subsegment()
```

Para crear un subsegmento para una función síncrona, utilice el decorador `@xray_recorder.capture`. Puede transferir un nombre para el subsegmento a la función de captura o dejarlo para utilizar el nombre de la función.

Example main.py: subsegmento de función

```
from aws_xray_sdk.core import xray_recorder

@xray_recorder.capture('## create_user')
def create_user():
...
```

Si se trata de una función asíncrona, utilice el decorador `@xray_recorder.capture_async` y pase un contexto asíncrono a la grabadora.

Example main.py: subsegmento de función asíncrona

```
from aws_xray_sdk.core.async_context import AsyncContext
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='my_service', context=AsyncContext())

@xray_recorder.capture_async('## create_user')
async def create_user():
    ...

async def main():
    await myfunc()
```

Cuando crea un subsegmento dentro de un segmento o de otro subsegmento, el SDK de X-Ray para Python genera un ID para dicho subsegmento y registra la hora de inicio y la hora de finalización.

Example Subsegmentos con metadatos

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
```

```
  },
```

Adición de anotaciones y metadatos a los segmentos con el SDK de X-Ray para Python

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede registrar información adicional acerca de las solicitudes, el entorno o su aplicación con anotaciones y metadatos. Puede añadir anotaciones y metadatos a los segmentos que crea el SDK de X-Ray o a los subsegmentos personalizados que cree usted mismo.

Las anotaciones son pares de clave-valor con valores de cadena, numéricos o booleanos. Las anotaciones se indexan para su uso con expresiones de filtro. Utilice anotaciones para registrar los datos que desee utilizar para agrupar rastros en la consola o cuando llame a la API de `GetTraceSummaries`.

Los metadatos son pares de clave-valor con valores de cualquier tipo, por ejemplo objetos y listas, pero que no se indexan para utilizarlos con expresiones de filtro. Utilice los metadatos para registrar datos adicionales que desee almacenar en el rastro, pero que no necesite usar para hacer búsquedas.

Además de anotaciones y metadatos, también puede registrar cadenas de ID de usuario en los segmentos. IDs Los usuarios se registran en un campo independiente en los segmentos y se indexan para usarlos en la búsqueda.

Secciones

- Registro de anotaciones con el SDK de X-Ray para Python
- Registro de metadatos con el SDK de X-Ray para Python
- Grabación del usuario IDs con el SDK de X-Ray para Python

Registro de anotaciones con el SDK de X-Ray para Python

Utilice anotaciones para registrar información sobre segmentos o subsegmentos que desee indexar para las búsquedas.

Requisitos de anotación

- Claves: la clave de una anotación de X-Ray puede contener hasta 500 caracteres alfanuméricos. No se pueden usar espacios ni símbolos, excepto el punto (.)
- Valores: el valor de una anotación de X-Ray puede contener hasta 1000 caracteres Unicode.
- Número de anotaciones: se pueden utilizar hasta 50 anotaciones por rastro.

Para registrar anotaciones

1. Obtenga una referencia al segmento o subsegmento actual desde `xray_recorder`.

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_segment()
   ```

   o

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_subsegment()
   ```

2. Llame a `put_annotation` con una clave de cadena y un valor booleano, numérico o de cadena.

   ```
   document.put_annotation("mykey", "my value");
   ```

   El siguiente ejemplo muestra cómo llamar a `putAnnotation` con una clave de cadena que incluye un punto y un valor booleano, numérico o de cadena.

   ```
   document.putAnnotation("testkey.test", "my value");
   ```

También puede utilizar el método `put_annotation` en el `xray_recorder`. Este método registra las anotaciones en el subsegmento actual o, si no hay ningún subsegmento abierto, en el segmento.

```
xray_recorder.put_annotation("mykey", "my value");
```

El SDK registra las anotaciones como pares de clave-valor en un objeto `annotations` del documento de segmento. Si llama dos veces a `put_annotation` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Para encontrar rastros que tengan anotaciones con valores específicos, utilice la palabra clave `annotation[key]` en una expresión de filtro.

Registro de metadatos con el SDK de X-Ray para Python

> **Warning**
>
> No agregue objetos con referencias circulares como valores de metadatos en el X-Ray SDK para Python. Estos objetos no se pueden serializar en JSON y pueden crear bucles infinitos en el SDK. Además, evite agregar objetos grandes y complejos como metadatos para evitar problemas de rendimiento.

Utilice los metadatos para registrar información sobre segmentos o subsegmentos que no necesite indexar para las búsquedas. Los valores de metadatos pueden ser cadenas, números, booleanos o cualquier objeto que se pueda serializar en un objeto o matriz JSON.

Para registrar metadatos

1. Obtenga una referencia al segmento o subsegmento actual desde `xray_recorder`.

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_segment()
   ```

   o

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_subsegment()
   ```

2. Llame a `put_metadata` con una clave de cadena; un valor booleano, numérico, de cadena o de objeto y un espacio de nombres de cadena.

```
document.put_metadata("my key", "my value", "my namespace");
```

o

Llame a `put_metadata` con solo una clave y un valor.

```
document.put_metadata("my key", "my value");
```

También puede utilizar el método `put_metadata` en el `xray_recorder`. Este método registra los metadatos en el subsegmento actual o, si no hay ningún subsegmento abierto, en el segmento.

```
xray_recorder.put_metadata("my key", "my value");
```

Si no especifica un espacio de nombres, el SDK utiliza `default`. Si llama dos veces a `put_metadata` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Grabación del usuario IDs con el SDK de X-Ray para Python

Registre IDs el usuario en los segmentos de solicitud para identificar al usuario que envió la solicitud.

Para registrar al usuario IDs

1. Obtenga una referencia al segmento actual desde `xray_recorder`.

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_segment()
   ```

2. Llame a `setUser` con un ID de cadena del usuario que envió la solicitud.

   ```
   document.set_user("U12345");
   ```

Puede llamar a `set_user` en sus controladores para registrar el ID de usuario en cuanto la aplicación empiece a procesar la solicitud.

Para buscar rastros de un ID de usuario, utilice la palabra clave `user` en una expresión de filtro.

Instrumentación de marcos de trabajo web implementados en entornos sin servidor

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de AWS X-Ray para Python admite la instrumentación de marcos web implementados en aplicaciones sin servidor. La arquitectura nativa de la nube carece de servidores, lo que le permite delegar más responsabilidades de gestión en AWS y mejorar la agilidad y la innovación.

La arquitectura sin servidor es un modelo de aplicaciones de software que permite crear y ejecutar aplicaciones y servicios sin preocuparse por los servidores. En esta arquitectura, ya no son necesarias las tareas de administración de la infraestructura, como el aprovisionamiento de servidores o clústeres, la aplicación de parches, el mantenimiento del sistema operativo y el aprovisionamiento de capacidad. Puede crear soluciones sin servidor para prácticamente cualquier tipo de aplicación o servicio de backend. Además, será usted quien se encargue de administrar todo lo necesario para ejecutar y escalar las aplicaciones con alta disponibilidad.

En este tutorial, se muestra cómo AWS X-Ray instrumentar automáticamente un marco web, como Flask o Django, que se implementa en un entorno sin servidor. La instrumentación de rayos X de la aplicación le permite ver todas las llamadas posteriores que se realizan, empezando por Amazon API Gateway y pasando por su AWS Lambda función, y las llamadas salientes que realiza la aplicación.

El SDK de X-Ray para Python admite los siguientes marcos de trabajo de aplicaciones de Python:

- Flask versión 0.8 o posteriores
- Django versión 1.0 o posteriores

En este tutorial se desarrolla un ejemplo de aplicación sin servidor que está implementada en Lambda y se invoca a través de API Gateway. En este tutorial se utiliza Zappa para implementar automáticamente la aplicación en Lambda y para configurar el punto de conexión de API Gateway.

Requisitos previos

- Zappa
- Python: versión 2.7 o 3.6.
- AWS CLI— Compruebe que AWS CLI está configurada con la cuenta Región de AWS en la que va a implementar la aplicación.
- Pip
- Virtualenv

Paso 1: crear un entorno

En este paso, va a crear un entorno virtual utilizando `virtualenv` para hospedar la aplicación.

1. Con el AWS CLI, cree un directorio para la aplicación. A continuación, cambie al nuevo directorio.

   ```
   mkdir serverless_application
   cd serverless_application
   ```

2. Cree un entorno virtual en este nuevo directorio. Utilice el comando siguiente para activarlo.

   ```
   # Create our virtual environment
   virtualenv serverless_env

   # Activate it
   source serverless_env/bin/activate
   ```

3. Instale X-Ray, Flask, Zappa y la biblioteca de solicitudes en el entorno.

   ```
   # Install X-Ray, Flask, Zappa, and Requests into your environment
   pip install aws-xray-sdk flask zappa requests
   ```

4. Añada el código de la aplicación al directorio `serverless_application`. En este caso, puede utilizar como referencia el ejemplo Hello World de Flasks.

En el directorio `serverless_application`, cree un archivo llamado `my_app.py`. A continuación, utilice un editor de texto para añadir los siguientes comandos. Esta aplicación instrumenta la biblioteca de solicitudes, aplica parches en el middleware de la aplicación de Flask y abre el punto de enlace `'/'`.

```
# Import the X-Ray modules
from aws_xray_sdk.ext.flask.middleware import XRayMiddleware
from aws_xray_sdk.core import patcher, xray_recorder
from flask import Flask
import requests

# Patch the requests module to enable automatic instrumentation
patcher.patch(('requests',))

app = Flask(__name__)

# Configure the X-Ray recorder to generate segments with our service name
xray_recorder.configure(service='My First Serverless App')

# Instrument the Flask application
XRayMiddleware(app, xray_recorder)

@app.route('/')
def hello_world():
    resp = requests.get("https://aws.amazon.com")
    return 'Hello, World: %s' % resp.url
```

Paso 2: Crear e implementar un entorno de Zappa

En este paso, utilizará Zappa para configurar automáticamente un punto de conexión de API Gateway e implementarlo en Lambda.

1. Inicialice Zappa desde dentro del directorio `serverless_application`. En este ejemplo, hemos utilizado la configuración predeterminada. Sin embargo, si tiene sus propias preferencias de personalización, Zappa le mostrará las instrucciones de configuración.

```
zappa init
```

```
What do you want to call this environment (default 'dev'): dev
```

```
...
What do you want to call your bucket? (default 'zappa-*******'): zappa-*******
...
...
It looks like this is a Flask application.
What's the modular path to your app's function?
This will likely be something like 'your_module.app'.
We discovered: my_app.app
Where is your app's function? (default 'my_app.app'): my_app.app
...
Would you like to deploy this application globally? (default 'n') [y/n/
(p)rimary]: n
```

2. Habilite X-Ray. Abra el archivo `zappa_settings.json` y compruebe que sea similar al del ejemplo.

```
{
    "dev": {
        "app_function": "my_app.app",
        "aws_region": "us-west-2",
        "profile_name": "default",
        "project_name": "serverless-exam",
        "runtime": "python2.7",
        "s3_bucket": "zappa-*********"
    }
}
```

3. Añada `"xray_tracing": true` como entrada en el archivo de configuración.

```
{
    "dev": {
        "app_function": "my_app.app",
        "aws_region": "us-west-2",
        "profile_name": "default",
        "project_name": "serverless-exam",
        "runtime": "python2.7",
        "s3_bucket": "zappa-*********",
        "xray_tracing": true
    }
}
```

4. Implemente la aplicación . Esto configurará automáticamente el punto de conexión de API Gateway y cargará el código en Lambda.

```
zappa deploy
```

```
...
Deploying API Gateway..
Deployment complete!: https://**********.execute-api.us-west-2.amazonaws.com/dev
```

Paso 3: habilitar el rastreo de X-Ray para API Gateway

En este paso, interactuará con la consola de API Gateway para habilitar el rastreo de X-Ray.

1. Inicie sesión en la consola de API Gateway Consola de administración de AWS y ábrala en https://console.aws.amazon.com/apigateway/.
2. Busque la API que acaba de generar. La URL debería parecerse a esta: `serverless-exam-dev`.
3. Elija Etapas.
4. Seleccione el nombre de la etapa de implementación. El valor predeterminado es `dev`.
5. En la pestaña Logs/Tracing (Registros/rastreo), active la casilla Enable X-Ray Tracing (Habilitar rastreo de X-Ray).
6. Elija Save changes (Guardar cambios).
7. Obtenga acceso al punto de enlace a través del navegador. Si utilizó la aplicación `Hello World` de ejemplo, debería aparecer lo siguiente.

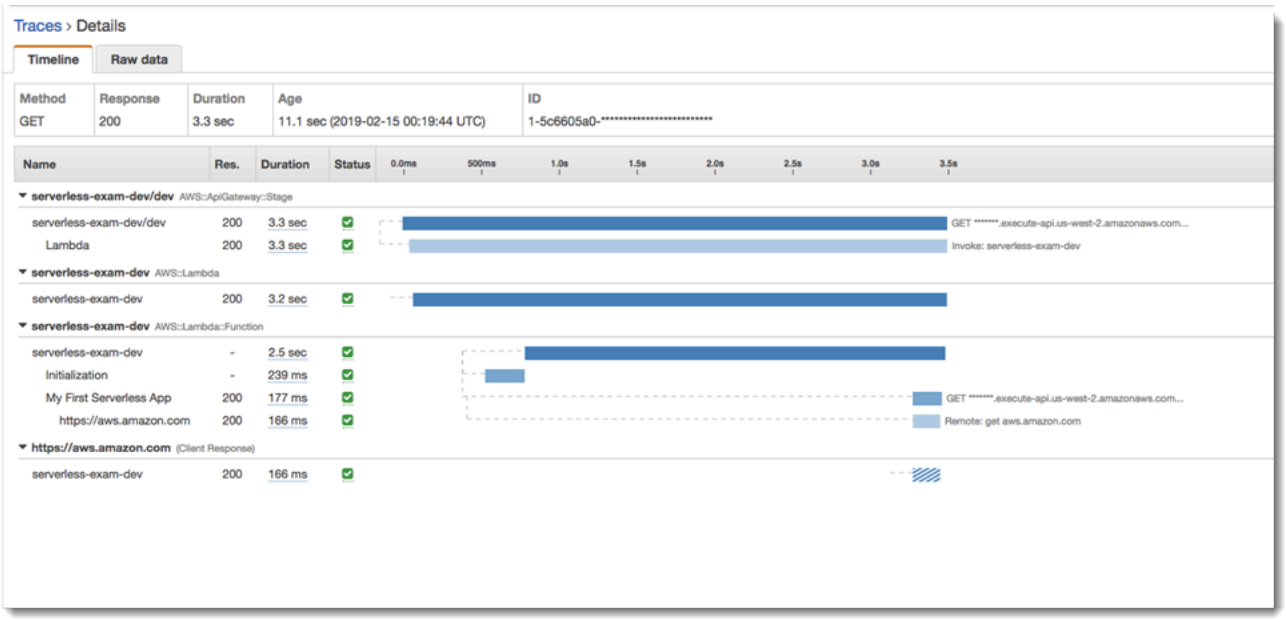
```
"Hello, World: https://aws.amazon.com/"
```

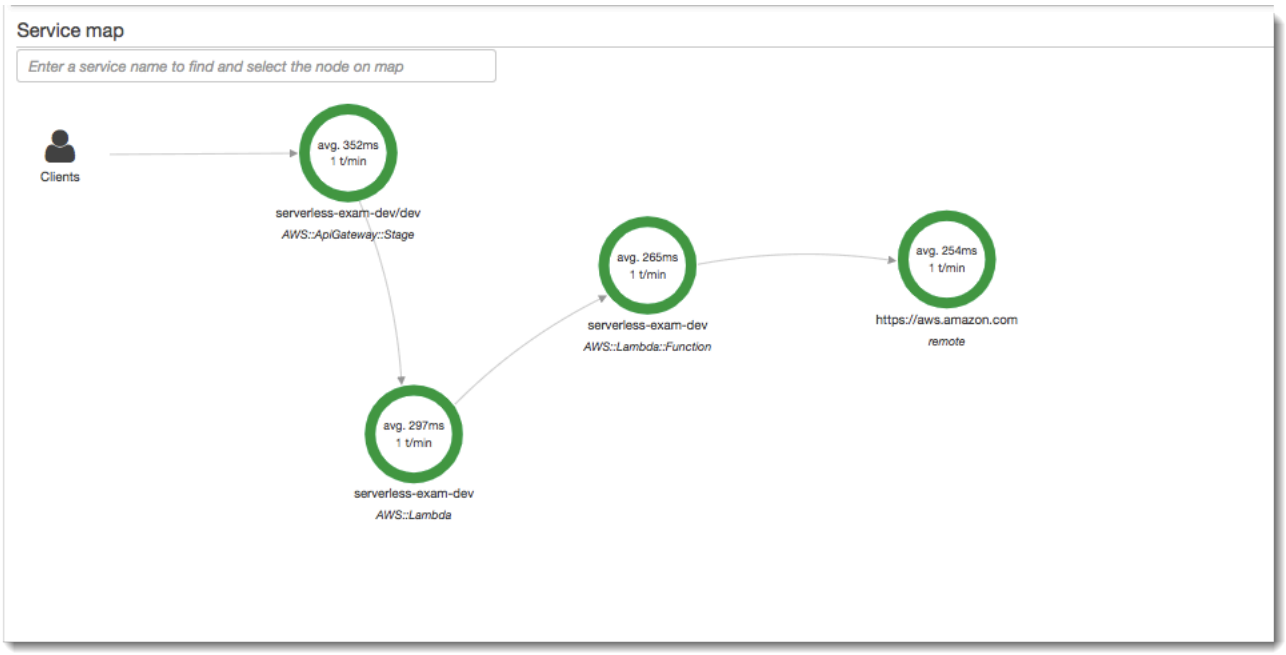
Paso 4: Consultar el registro de seguimiento creado

En este paso, interactuará con la consola de X-Ray para ver el rastro creado por la aplicación de ejemplo. Para ver una explicación detallada del análisis del seguimiento, consulte Uso del mapa de servicio.

1. Inicie sesión en la consola de X-Ray Consola de administración de AWS y ábrala en https://console.aws.amazon.com/xray/casa.
2. Consulte los segmentos generados por API Gateway, la función de Lambda y el contenedor de Lambda.

3. En el segmento de la función de Lambda, consulte un subsegmento llamado `My First Serverless App`. Detrás hay un segundo subsegmento llamado `https://aws.amazon.com`.
4. Durante la inicialización, Lambda podría generar también un tercer subsegmento llamado `initialization`.





Paso 5: Eliminar

Termine siempre los recursos que ya no utilice para evitar que se acumulen costos inesperados. Tal y como se explica en este tutorial, existen herramientas como Zappa que optimizan la implementación sin servidor.

Para eliminar la aplicación de Lambda, API Gateway y Amazon S3, ejecute el siguiente comando en el directorio del proyecto a través de la AWS CLI.

```
zappa undeploy dev
```

Siguientes pasos

Añada más funciones a su aplicación añadiendo AWS clientes e instrumentándolos con X-Ray. Consulte más información sobre las opciones de computación sin servidor de Sin servidor en AWS.

Uso de .NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede instrumentar su aplicación .NET de dos maneras para enviar rastros a X-Ray:

- AWS Distro for OpenTelemetry .NET: una AWS distribución que proporciona un conjunto de bibliotecas de código abierto para enviar métricas y rastreos correlacionados a varias soluciones de AWS monitoreo, incluidas Amazon y Amazon OpenSearch Service CloudWatch AWS X-Ray, a través de AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK para .NET: conjunto de bibliotecas para generar y enviar trazas a X-Ray mediante el daemon X-Ray.

Para obtener más información, consulte Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs.

AWS Distro para OpenTelemetry .NET

Con AWS Distro para OpenTelemetry .NET, puede instrumentar sus aplicaciones una vez y enviar métricas y rastros correlacionados a varias soluciones de supervisión de AWS, incluidas Amazon CloudWatch, AWS X-Ray y Amazon OpenSearch Service. El uso de X-Ray con AWS Distro para OpenTelemetry requiere dos componentes: un SDK de OpenTelemetry habilitado para usarlo con X-Ray y el AWS Distro para OpenTelemetry Collector habilitado para usarlo con X-Ray.

Para empezar, consulte la documentación de AWSDistro para OpenTelemetry .NET.

Para obtener más información sobre el uso de AWS Distro para OpenTelemetry con AWS X-Ray y otros Servicios de AWS, consulte AWS Distro para OpenTelemetry o la documentación de AWS Distro para OpenTelemetry.

Para obtener información adicional sobre los lenguajes compatibles y el uso, consulte AWS Observability en GitHub.

AWS X-Ray SDK para.NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El X-Ray SDK para .NET es una biblioteca para instrumentar aplicaciones web de C# .NET, aplicaciones web .NET Core y funciones de .NET Core. AWS Lambda Proporciona clases y métodos para generar y enviar datos de rastro al daemon de X-Ray. Incluye información sobre las solicitudes entrantes atendidas por la aplicación y las llamadas que la aplicación realiza a bases de datos descendentes Servicios de AWS APIs, web HTTP y SQL.

> **Note**
>
> El SDK de X-Ray para .NET es un proyecto de código abierto. Puedes seguir el proyecto y enviar las incidencias y solicitudes de cambios en GitHub: github. com/aws/aws- xray-sdk-dotnet

Para las aplicaciones web, comience por añadir un controlador de mensajes a su configuración web para realizar el seguimiento de las solicitudes entrantes. El controlador de mensajes crea un segmento para cada solicitud rastreada y lo completa cuando se envía la respuesta. Mientras el segmento está abierto, puede utilizar los métodos del cliente del SDK para añadir información

al segmento y crear subsegmentos para rastrear llamadas posteriores. El SDK también registra automáticamente las excepciones que produce su aplicación mientras el segmento está abierto.

En el caso de las funciones de Lambda llamadas por una aplicación o un servicio instrumentados, Lambda lee el encabezado de rastreo y rastrea automáticamente las solicitudes muestreadas. Para otras funciones, puede configurar Lambda con el fin de muestrear y rastrear las solicitudes entrantes. En cualquier caso, Lambda crea el segmento y se lo proporciona al SDK de X-Ray.

> **Note**
>
> En Lambda, el SDK de X-Ray es opcional. Si no lo usa en su función, el mapa de servicio seguirá incluyendo un nodo para el servicio de Lambda y uno para cada función de Lambda. Al añadir el SDK, puede instrumentar el código de función para añadir subsegmentos al segmento de función registrado por Lambda. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

A continuación, use el SDK de X-Ray para .NET con el fin de instrumentar a sus clientes del AWS SDK for .NET. Cada vez que realizas una llamada a un servicio intermedio Servicio de AWS o a un recurso con un cliente instrumentado, el SDK registra la información sobre la llamada en un subsegmento. AWS los servicios y los recursos a los que accedes dentro de los servicios aparecen como nodos descendentes en el mapa de rastreo para ayudarte a identificar los errores y los problemas de limitación en las conexiones individuales.

El X-Ray SDK para .NET también proporciona instrumentación para llamadas posteriores a bases de datos HTTP web APIs y SQL. El método de extensión `GetResponseTraced` para `System.Net.HttpWebRequest` rastrea llamadas HTTP salientes. Puede utilizar la versión de `SqlCommand` del SDK de X-Ray para .NET para instrumentar consultas SQL.

En cuanto empiece a utilizar el SDK, personalice su comportamiento configurando la grabadora y el controlador de mensajes. Puede añadir complementos para registrar los datos sobre los recursos informáticos que ejecutan su aplicación, personalizar el comportamiento de muestreo mediante la definición de reglas de muestreo y definir el nivel de log para ver más o menos información del SDK en los logs de las aplicaciones.

Registre información adicional acerca de las solicitudes y el trabajo que la aplicación realiza en anotaciones y metadatos. Las anotaciones son pares sencillos de clave-valor que se indexan para su uso con expresiones de filtro para poder buscar rastros que contengan datos específicos. Las

entradas de metadatos son menos restrictivas y pueden registrar objetos y matrices completos, es decir, todo lo que se pueda serializar en JSON.

> **Anotaciones y metadatos**
>
> Las anotaciones y los metadatos son texto arbitrario que se agrega a los segmentos con el SDK de X-Ray. Las anotaciones se indexan para su uso con expresiones de filtro. Los metadatos no se indexan pero se pueden ver en el segmento sin procesar con la consola o la API de X-Ray. Cualquier persona a la que conceda acceso de lectura a X-Ray puede ver estos datos.

Cuando tenga muchos clientes instrumentados en su código, un único segmento de solicitud puede contener un gran número de subsegmentos, uno para cada llamada realizada con un cliente instrumentado. Puede organizar y agrupar los subsegmentos incluyendo las llamadas del cliente en subsegmentos personalizados. Puede crear un subsegmento personalizado para una función completa o para cualquier sección de código, y registrar los metadatos y las anotaciones en el subsegmento en lugar de escribirlo todo en el segmento principal.

Para acceder a documentos de referencia sobre las clases y los métodos de SDK, consulte lo siguiente:

- AWS X-Ray Referencia de la API de SDK for .NET
- AWS X-Ray Referencia de API de SDK for .NET Core

El mismo paquete admite tanto .NET como .NET Core, pero las clases que se utilizan varían. Los ejemplos de este capítulo contienen un enlace a la referencia del API de .NET, a menos que la clase sea específica de .NET Core.

Requisitos

El X-Ray SDK para .NET requiere el.NET Framework 4.5 o una versión posterior AWS SDK for .NET y.

Para las aplicaciones y funciones de .NET Core, el SDK requiere .NET Core 2.0 o posterior.

Integración del SDK de X-Ray para .NET en su aplicación

NuGet Utilícelo para agregar el X-Ray SDK para .NET a su aplicación.

Para instalar el SDK de X-Ray para .NET con el administrador de NuGet paquetes de Visual Studio

1. Elija Tools, NuGet Package Manager y Manage NuGet Packages for Solution.
2. Busque AWSXRayRecorder.
3. Elija el paquete y, a continuación, haga clic en Install (Instalar).

Administración de dependencias

El SDK de X-Ray para .NET está disponible en Nuget. Instale el SDK con el administrador de paquetes.

```
Install-Package AWSXRayRecorder -Version 2.10.1
```

El paquete NuGet `AWSXRayRecorder v2.10.1` tiene las siguientes dependencias:

NET Framework 4.5

```
AWSXRayRecorder (2.10.1)
|
|-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- AWSSDK.Core (>= 3.3.25.1)
|
|-- AWSXRayRecorder.Handlers.AspNet (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.AwsSdk (>= 2.8.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.EntityFramework (>= 1.1.1)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- EntityFramework (>= 6.2.0)
|
|-- AWSXRayRecorder.Handlers.SqlServer (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.System.Net (>= 2.7.3)
    |-- AWSXRayRecorder.Core (>= 2.10.1)
```

NET Framework 2.0

```
AWSXRayRecorder (2.10.1)
|
|-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- AWSSDK.Core (>= 3.3.25.1)
|   |-- Microsoft.AspNetCore.Http (>= 2.0.0)
|   |-- Microsoft.Extensions.Configuration (>= 2.0.0)
|   |-- System.Net.Http (>= 4.3.4)
|
|-- AWSXRayRecorder.Handlers.AspNetCore (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- Microsoft.AspNetCore.Http.Extensions (>= 2.0.0)
|   |-- Microsoft.AspNetCore.Mvc.Abstractions (>= 2.0.0)
|
|-- AWSXRayRecorder.Handlers.AwsSdk (>= 2.8.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.EntityFramework (>= 1.1.1)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- Microsoft.EntityFrameworkCore.Relational (>= 3.1.0)
|
|-- AWSXRayRecorder.Handlers.SqlServer (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- System.Data.SqlClient (>= 4.4.0)
|
|-- AWSXRayRecorder.Handlers.System.Net (>= 2.7.3)
    |-- AWSXRayRecorder.Core (>= 2.10.1)
```

Para obtener más información sobre la administración de dependencias, consulte la documentación de Microsoft sobre la dependencia de NuGet y la resolución de dependencias de paquetes NuGet.

Configuración del SDK de X-Ray para .NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y

Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede configurar el SDK de X-Ray para .NET mediante el uso de complementos a fin de incluir información sobre el servicio que sus aplicaciones ejecutan, modificar la conducta predeterminada de muestreo o agregar reglas de muestreo que se aplican a las solicitudes dirigidas a rutas específicas.

Para las aplicaciones web .NET, añada claves a la sección `appSettings` del archivo `Web.config`.

Example Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Para .NET Core, cree un archivo denominado `appsettings.json` con una clave de nivel superior llamada `XRay`.

Example.NET appsettings.json

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin",
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

A continuación, en el código de la aplicación, cree un objeto de configuración y utilícelo para inicializar la grabadora de X-Ray. Hágalo antes de inicializar la grabadora.

Example.NET Core Program.cs: configuración de grabadora

```
using Amazon.XRay.Recorder.Core;
...
```

```
AWSXRayRecorder.InitializeInstance(configuration);
```

Si está instrumentando una aplicación web de .NET Core, también puede pasar el objeto de configuración al método `UseXRay` al configurar el controlador de mensajes. Para las funciones de Lambda utilice el método `InitializeInstance` como se muestra más arriba.

Para obtener más información acerca del API de configuración de .NET Core, consulte Configurar una aplicación ASP.NET Core en docs.microsoft.com.

Secciones

- Plugins
- Reglas de muestreo
- Registro (.NET)
- Registro (.NET Core)
- Variables de entorno

Plugins

Utilice complementos para agregar datos sobre el servicio que aloja su aplicación.

Plugins

- Amazon EC2 : `EC2Plugin` agrega el ID de la instancia, la zona de disponibilidad y el grupo de CloudWatch registros.
- Elastic Beanstalk: `ElasticBeanstalkPlugin` añade el nombre de entorno, la etiqueta de versión y el ID de implementación.
- Amazon ECS: `ECSPlugin` agrega el ID de contenedor.

Para utilizar un complemento, configure el cliente del SDK de X-Ray para .NET añadiendo el ajuste `AWSXRayPlugins`. Si hay varios complementos para su aplicación, especifíquelos todos en la misma configuración, separados por comas.

Example Web.config - complementos

```
<configuration>
  <appSettings>
```

```
    <add key="AWSXRayPlugins" value="EC2Plugin,ElasticBeanstalkPlugin"/>
  </appSettings>
</configuration>
```

Example.NET Core appsettings.json: complementos

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin,ElasticBeanstalkPlugin"
  }
}
```

Reglas de muestreo

El SDK utiliza las reglas de muestreo que define el usuario en la consola de X-Ray para determinar qué solicitudes registrar. La regla predeterminada rastrea la primera solicitud cada segundo y el 5 % de las solicitudes adicionales de todos los servicios que envían rastros a X-Ray. Cree reglas adicionales en la consola de X-Ray para personalizar la cantidad de datos registrados para cada una de sus aplicaciones.

El SDK aplica las reglas personalizadas en el orden en que se definen. Si una solicitud coincide con varias reglas personalizadas, el SDK solo aplica la primera regla.

> **Note**
>
> Si el SDK no puede comunicarse con X-Ray para obtener las reglas de muestreo, vuelve a una regla local predeterminada de la primera solicitud cada segundo y del 5 % de las solicitudes adicionales por host. Esto puede ocurrir si el host no tiene permiso para llamar al muestreo APIs o no puede conectarse al daemon X-Ray, que actúa como proxy TCP para las llamadas a la API realizadas por el SDK.

También puede configurar el SDK para que cargue las reglas de muestreo desde un documento JSON. El SDK puede usar las reglas locales como respaldo para los casos en que el muestreo de X-Ray no esté disponible, o puede usar las reglas locales exclusivamente.

Example sampling-rules.json

```
{
```

```
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

En este ejemplo se define una regla personalizada y una regla predeterminada. La regla personalizada aplica un porcentaje de muestreo del 5 % sin un número mínimo de solicitudes de rastreo para las rutas incluidas bajo `/api/move/`. La regla predeterminada rastrea la primera solicitud cada segundo y el 10 % de las solicitudes adicionales.

La desventaja de definir las reglas localmente es que el objetivo establecido lo aplica cada instancia de la grabadora de forma independiente, en lugar de ser administrado por el servicio de X-Ray. A medida que se implementan más hosts, el porcentaje establecido se multiplica, lo que dificulta el control de la cantidad de datos registrados.

Sí AWS Lambda, no puedes modificar la frecuencia de muestreo. Si un servicio instrumentado llama a su función, Lambda registrará las llamadas que generaron solicitudes muestreadas por ese servicio. Si el rastreo activo está activado y no hay ningún encabezado de rastreo, Lambda toma la decisión de muestreo.

Para configurar reglas de respaldo, indique al SDK de X-Ray para .NET que cargue reglas de muestreo desde un archivo con la configuración `SamplingRuleManifest`.

Example.NET Web.config: reglas de muestreo

```
<configuration>
  <appSettings>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
```

```
</configuration>
```

Example.NET Core appsettings.json: reglas de muestreo

```
{
  "XRay": {
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

Para utilizar solo reglas locales, cree la grabadora con una `LocalizedSamplingStrategy`. Si tiene reglas de copia de seguridad configuradas, elimine dicha configuración.

Example.NET global.asax: reglas de muestreo locales

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("samplingrules.json")).Build();
AWSXRayRecorder.InitializeInstance(recorder: recorder);
```

Example.NET Core Program.cs: reglas de muestreo locales

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("sampling-rules.json")).Build();
AWSXRayRecorder.InitializeInstance(configuration,recorder);
```

Registro (.NET)

El SDK de X-Ray para .NET usa el mismo mecanismo de registro que AWS SDK for .NET. Si ya configuró la aplicación para registrar la AWS SDK for .NET salida, la misma configuración se aplica a la salida del X-Ray SDK for .NET.

Para configurar el registro, añada una sección de configuración denominada `aws` al archivo `App.config` o `Web.config`.

Example Web.config: registro

```
...
<configuration>
  <configSections>
```

```
    <section name="aws" type="Amazon.AWSSection, AWSSDK.Core"/>
  </configSections>
  <aws>
    <logging logTo="Log4Net"/>
  </aws>
</configuration>
```

Para obtener más información, consulte Configuración del AWS SDK for .NET para su aplicación en la Guía para desarrolladores del AWS SDK for .NET .

Registro (.NET Core)

El SDK de X-Ray para .NET usa las mismas opciones de registro que AWS SDK for .NET. Para configurar el registro para las aplicaciones .NET Core, transfiera la opción de registro al método `AWSXRayRecorder.RegisterLogger`.

Por ejemplo, para utilizar log4net, cree un archivo de configuración que defina el registrador, el formato de salida y la ubicación del archivo.

Example.NET Core log4net.config

```
<?xml version="1.0" encoding="utf-8" ?>
<log4net>
  <appender name="FileAppender" type="log4net.Appender.FileAppender,log4net">
    <file value="c:\logs\sdk-log.txt" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date [%thread] %level %logger - %message%newline" />
    </layout>
  </appender>
  <logger name="Amazon">
    <level value="DEBUG" />
    <appender-ref ref="FileAppender" />
  </logger>
</log4net>
```

A continuación, cree el registrador y aplique la configuración en el código del programa.

Example.NET Core Program.cs: registro

```
using log4net;
using Amazon.XRay.Recorder.Core;
```

```
class Program
{
  private static ILog log;
  static Program()
  {
    var logRepository = LogManager.GetRepository(Assembly.GetEntryAssembly());
    XmlConfigurator.Configure(logRepository, new FileInfo("log4net.config"));
    log = LogManager.GetLogger(typeof(Program));
    AWSXRayRecorder.RegisterLogger(LoggingOptions.Log4Net);
  }
  static void Main(string[] args)
  {
  ...
  }
}
```

Para obtener más información sobre cómo configurar log4net, consulte Configuration en logging.apache.org.

Variables de entorno

Puede usar variables de entorno para configurar el SDK de X-Ray para .NET. El SDK admite las siguientes variables.

- `AWS_XRAY_TRACING_NAME`: establezca el nombre de servicio que el SDK utiliza para los segmentos. Anula el nombre de servicio que se ha establecido en la estrategia de nomenclatura de segmento del filtro de servlet.
- `AWS_XRAY_DAEMON_ADDRESS`: establezca el host y el puerto del oyente del daemon de X-Ray. De forma predeterminada, el SDK utiliza `127.0.0.1:2000` tanto para los datos de rastro (UDP) como para el muestreo (TCP). Use esta variable si ha configurado el daemon para que escuche en un puerto diferente o si se ejecuta en un host diferente.

  Formato

  - El mismo puerto: *`address`*`:`*`port`*
  - Puertos diferentes: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_CONTEXT_MISSING`: establezca esta opción en `RUNTIME_ERROR` para generar excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.

Valores válidos

- `RUNTIME_ERROR`: lance una excepción de tiempo de ejecución.
- `LOG_ERROR`: registre un error y continúe (predeterminado).
- `IGNORE_ERROR`: ignore el error y continúe.

Se pueden producir errores relativos a segmentos o subsegmentos inexistentes al intentar usar un cliente instrumentado en el código de inicio que se ejecuta cuando no hay ninguna solicitud abierta o en el código que inicia un nuevo subproceso.

Instrumentación de las solicitudes HTTP entrantes con el SDK de X-Ray para .NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede usar el SDK de X-Ray para rastrear las solicitudes HTTP entrantes que su aplicación sirve en una EC2 instancia de Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Utilice un controlador de mensajes para instrumentar las solicitudes HTTP entrantes. Cuando agrega el controlador de mensajes de X-Ray a su aplicación, el SDK de X-Ray para .NET crea un segmento para cada solicitud muestreada. Este segmento incluye el momento, el método y la disposición de la solicitud HTTP. La instrumentación adicional crea subsegmentos en este segmento.

> **Note**
>
> En el AWS Lambda caso de las funciones, Lambda crea un segmento para cada solicitud muestreada. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

Cada segmento tiene un nombre que identifica la aplicación en el mapa de servicio. El nombre del segmento se puede asignar de forma estática o se puede configurar el SDK para que le asigne un nombre dinámico en función del encabezado del host de la solicitud entrante. La nomenclatura dinámica permite agrupar los rastros en función del nombre de dominio de la solicitud y aplicar un nombre predeterminado si el nombre no coincide con el patrón esperado (por ejemplo, si el encabezado del host está falsificado).

> **Solicitudes reenviadas**
>
> Si un equilibrador de carga u otro intermediario reenvía una solicitud a la aplicación, X-Ray toma la IP de cliente del encabezado `X-Forwarded-For` de la solicitud en lugar de tomar la IP de origen del paquete IP. La IP de cliente que se graba para una solicitud reenviada puede estar falsificada, por lo que no se debe confiar en ella.

El controlador de mensajes crea un segmento para cada solicitud entrante con un bloque `http` que contiene la siguiente información:

- Método HTTP: GET, POST, PUT, DELETE, etc.
- Dirección del cliente: la dirección IP del cliente que envió la solicitud.
- Código de respuesta: el código de respuesta HTTP para la solicitud finalizada.
- Intervalo: la hora de inicio (cuando se recibió la solicitud) y la hora de finalización (cuando se envió la respuesta).
- Agente del usuario: el `user-agent` de la solicitud.
- Longitud del contenido: la `content-length` de la respuesta.

Secciones

Instrumentación de las solicitudes entrantes (.NET)

Para instrumentar las solicitudes que atiende su aplicación, llame a `RegisterXRay` en el método `Init` del archivo `global.asax`.

Example global.asax: controlador de mensajes

```
using System.Web.Http;
using Amazon.XRay.Recorder.Handlers.AspNet;

namespace SampleEBWebApplication
{
  public class MvcApplication : System.Web.HttpApplication
  {
    public override void Init()
    {
      base.Init();
      AWSXRayASPNET.RegisterXRay(this, "MyApp");
    }
  }
}
```

Instrumentación de las solicitudes entrantes (.NET Core)

Para instrumentar las solicitudes atendidas por la aplicación, llame al método `UseXRay` antes que a cualquier otro middleware del método `Configure` de su clase Startup, ya que lo ideal es que el middleware X-Ray sea el primer middleware en procesar la solicitud y el último en procesar la respuesta en el proceso.

> **Note**
>
> En el caso de .NET Core 2.0, si tiene un método `UseExceptionHandler` en la aplicación, asegúrese de llamar a `UseXRay` después del método `UseExceptionHandler` para asegurarse de que se registren las excepciones.

Example Startup.cs

<caption>.NET Core 2.1 and above</caption>

```
using Microsoft.AspNetCore.Builder;

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {
    app.UseXRay("MyApp");
    // additional middleware
```

```
    ...
  }
```

<caption>.NET Core 2.0</caption>

```
using Microsoft.AspNetCore.Builder;

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {
    app.UseExceptionHandler("/Error");
    app.UseXRay("MyApp");
    // additional middleware
    ...
  }
```

El método `UseXRay` también puede tomar un objeto de configuración como segundo argumento.

```
app.UseXRay("MyApp", configuration);
```

Configuración de una estrategia de nomenclatura de segmentos

AWS X-Ray utiliza un nombre de servicio para identificar su aplicación y distinguirla del resto de aplicaciones, bases de datos, AWS recursos externos APIs y que utiliza su aplicación. Cuando el SDK de X-Ray genera segmentos para las solicitudes entrantes, registra el nombre del servicio de la aplicación en el campo de nombre del segmento.

El SDK de X-Ray puede nombrar los segmentos utilizando el nombre de host en el encabezado de la solicitud HTTP. Sin embargo, este encabezado se puede falsificar, lo que podría provocar nodos inesperados en el mapa de servicio. Para evitar que el SDK nombre los segmentos de forma incorrecta debido a que las solicitudes tienen encabezados de host falsificados, debe especificar un nombre predeterminado para las solicitudes entrantes.

Si la aplicación atiende solicitudes de varios dominios, puede configurar el SDK para que utilice una estrategia de nomenclatura dinámica que refleje esto en los nombres de los segmentos. Una estrategia de nomenclatura dinámica permite al SDK usar el nombre de host para las solicitudes que coinciden con un patrón esperado y aplicar el nombre predeterminado a las solicitudes que no coincidan.

Por ejemplo, es posible que tenga una sola aplicación que atienda solicitudes a tres subdominios: `www.example.com`, `api.example.com` y `static.example.com`. Puede usar una estrategia

de nomenclatura dinámica con el patrón `*.example.com` para identificar los segmentos de cada subdominio con un nombre diferente, lo que da como resultado tres nodos de servicio en el mapa de servicio. Si su aplicación recibe solicitudes con un nombre de host que no coincide con el patrón, verá un cuarto nodo en el mapa de servicio con el nombre alternativo que especifique.

Para utilizar el mismo nombre para todos los segmentos de solicitud, especifique el nombre de la aplicación cuando inicialice el controlador de mensajes, tal y como se muestra en la sección anterior. Esto tiene el mismo efecto que crear un `FixedSegmentNamingStrategy` y pasárselo al método `RegisterXRay`.

```
AWSXRayASPNET.RegisterXRay(this, new FixedSegmentNamingStrategy("MyApp"));
```

> **Note**
>
> Puede anular el nombre de servicio predeterminado que ha definido en el código mediante la `AWS_XRAY_TRACING_NAME`variable de entorno???.

Una estrategia de nomenclatura dinámica define un patrón con el que deben coincidir los nombres de host y un nombre predeterminado que se utiliza si el nombre de host de la solicitud HTTP no coincide con el patrón. Para asignar nombres a los segmentos dinámicamente, cree un objeto `DynamicSegmentNamingStrategy` y páseselo al método `RegisterXRay`.

```
AWSXRayASPNET.RegisterXRay(this, new DynamicSegmentNamingStrategy("MyApp",
 "*.example.com"));
```

Rastreo de llamadas AWS del SDK con el X-Ray SDK para .NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando la aplicación realiza llamadas Servicios de AWS para almacenar datos, escribir en una cola o enviar notificaciones, el X-Ray SDK for .NET rastrea las llamadas en sentido descendente en subsegmentos. El rastreo Servicios de AWS y los recursos a los que accede dentro de esos servicios (por ejemplo, un bucket de Amazon S3 o una cola de Amazon SQS) aparecen como nodos descendentes en el mapa de rastreo de la consola X-Ray.

Puede instrumentar a todos sus AWS SDK for .NET clientes llamando `RegisterXRayForAllServices` antes de crearlos.

Example SampleController.cs: instrumentación de cliente de DynamoDB

```
using Amazon;
using Amazon.Util;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.AwsSdk;

namespace SampleEBWebApplication.Controllers
{
  public class SampleController : ApiController
  {
    AWSSDKHandler.RegisterXRayForAllServices();
    private static readonly Lazy<AmazonDynamoDBClient> LazyDdbClient = new
 Lazy<AmazonDynamoDBClient>(() =>
    {
      var client = new AmazonDynamoDBClient(EC2InstanceMetadata.Region ??
 RegionEndpoint.USEast1);
      return client;
    });
```

Si desea instrumentar clientes para algunos servicios y no para otros, llame a `RegisterXRay` en lugar de `RegisterXRayForAllServices`. Reemplace el texto resaltado por el nombre de la interfaz de cliente del servicio.

```
AWSSDKHandler.RegisterXRay<IAmazonDynamoDB>()
```

Para todos los servicios, puede ver el nombre de la API a la que se llama en la consola de X-Ray. Para un subconjunto de servicios, el SDK de X-Ray agrega información al segmento para proporcionar una mayor granularidad en el mapa de servicio.

Por ejemplo, cuando realiza una llamada con un cliente instrumentado de DynamoDB, el SDK agrega el nombre de tabla al segmento para las llamadas que se dirigen a una tabla. En la consola, cada tabla aparece como nodo independiente en el mapa de servicio, con un nodo genérico de DynamoDB para las llamadas que no se dirigen a una tabla.

Example Subsegmento para una llamada a DynamoDB con el fin de guardar un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Cuando accede a recursos designados, las llamadas a los siguientes servicios crean nodos adicionales en el mapa de servicio. Las llamadas que no están dirigidas a recursos concretos crean un nodo genérico en el servicio.

- Amazon DynamoDB: nombre de tabla
- Amazon Simple Storage Service: nombre de bucket y de clave
- Amazon Simple Queue Service: nombre de cola

Rastreo de llamadas a servicios web HTTP posteriores con el SDK de X-Ray para .NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando tu aplicación realiza llamadas a microservicios o a HTTP públicos APIs, puedes usar el método de `GetResponseTraced` extensión del SDK de X-Ray para .NET `System.Net.HttpWebRequest` para instrumentar esas llamadas y añadir la API al gráfico del servicio como un servicio descendente.

Example HttpWebRequest

```
using System.Net;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.System.Net;

private void MakeHttpRequest()
{
  HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://names.example.com/
api");
  request.GetResponseTraced();
}
```

Para las llamadas asíncronas, utilice `GetAsyncResponseTraced`.

```
request.GetAsyncResponseTraced();
```

Si utiliza `system.net.http.httpclient`, utilice el controlador de delegación `HttpClientXRayTracingHandler` para registrar las llamadas.

Example HttpClient

```
using System.Net.Http;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.System.Net;

private void MakeHttpRequest()
{
  var httpClient = new HttpClient(new HttpClientXRayTracingHandler(new
 HttpClientHandler()));
  httpClient.GetAsync(URL);
}
```

Cuando se instrumenta una llamada a un API web posterior, el del SDK de X-Ray para .NET registra un subsegmento con información sobre la solicitud HTTP y la respuesta. X-Ray utiliza el subsegmento para generar un segmento inferido para la API.

Example Subsegmento para una llamada HTTP posterior

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example Segmento inferido para una llamada HTTP posterior

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
```

```
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

Rastreo de consultas SQL con el SDK de X-Ray para .NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para .NET proporciona una clase de encapsulamiento `System.Data.SqlClient.SqlCommand` denominada `TraceableSqlCommand` que puede utilizar en lugar de `SqlCommand`. Puede inicializar un comando SQL con la clase `TraceableSqlCommand`.

Seguimiento de consultas SQL con métodos síncronos y asíncronos

Los siguientes ejemplos muestran cómo utilizar `TraceableSqlCommand` para rastrear automáticamente las consultas de SQL Server de forma síncrona y asíncrona.

Example **`Controller.cs`**: instrumentación de cliente de SQL (síncrono)

```
using Amazon;
using Amazon.Util;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.SqlServer;

private void QuerySql(int id)
{
  var connectionString = ConfigurationManager.AppSettings["RDS_CONNECTION_STRING"];
  using (var sqlConnection = new SqlConnection(connectionString))
  using (var sqlCommand = new TraceableSqlCommand("SELECT " + id, sqlConnection))
  {
    sqlCommand.Connection.Open();
    sqlCommand.ExecuteNonQuery();
  }
}
```

Puede ejecutar la consulta de forma asíncrona utilizando el método `ExecuteReaderAsync`.

Example **`Controller.cs`**: instrumentación de clientes de SQL (asíncronos)

```
using Amazon;
using Amazon.Util;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.SqlServer;
private void QuerySql(int id)
{
  var connectionString = ConfigurationManager.AppSettings["RDS_CONNECTION_STRING"];
  using (var sqlConnection = new SqlConnection(connectionString))
  using (var sqlCommand = new TraceableSqlCommand("SELECT " + id, sqlConnection))
  {
    await sqlCommand.ExecuteReaderAsync();
  }
}
```

Recopilación de consultas SQL realizadas a SQL Server

Puede habilitar la captura de `SqlCommand.CommandText` como parte del subsegmento creado por la consulta SQL. `SqlCommand.CommandText` aparece como el campo `sanitized_query` en el JSON del subsegmento. De forma predeterminada, esta característica está deshabilitada por motivos de seguridad.

> **Note**
>
> No habilite la característica de recopilación si está incluyendo información confidencial como texto sin cifrar en sus consultas SQL.

Puede habilitar la recopilación de consultas SQL de dos formas:

- Establezca la propiedad `CollectSqlQueries` en `true` en la configuración global de su aplicación.
- Establezca el parámetro `collectSqlQueries` de la instancia `TraceableSqlCommand` en `true` para recopilar llamadas dentro de la instancia.

Habilite la propiedad global CollectSqlQueries

Los siguientes ejemplos muestran cómo habilitar la propiedad `CollectSqlQueries` para .NET y .NET Core.

.NET

Para establecer la propiedad `CollectSqlQueries` en `true` en la configuración global de su aplicación en .NET, modifique el `appsettings` de su archivo `Web.config` o `App.config`, tal y como se muestra.

Example **`App.config`** o bien **`Web.config`**: habilitación global de la recopilación de consultas SQL

```
<configuration>
<appSettings>
    <add key="CollectSqlQueries" value="true">
</appSettings>
</configuration>
```

.NET Core

Para establecer la propiedad `CollectSqlQueries` en `true` en la configuración global de su aplicación en .NET Core, modifique el archivo `appsettings.json` en la clave de X-Ray, tal y como se muestra.

Example **`appsettings.json`**: habilitación global de la recopilación de consultas SQL

```
{
  "XRay": {
    "CollectSqlQueries":"true"
  }
}
```

Habilite el collectSqlQueries parámetro

Puede establecer el parámetro `collectSqlQueries` en la instancia `TraceableSqlCommand` en `true` para recopilar el texto de consulta SQL para las consultas de SQL Server realizadas con esa instancia. Si se establece el parámetro en `false` se deshabilita la característica `CollectSqlQuery` para la instancia `TraceableSqlCommand`.

> **Note**
>
> El valor de `collectSqlQueries` en la instancia `TraceableSqlCommand` anula el valor establecido en la configuración global de la propiedad `CollectSqlQueries`.

Example Ejemplo **`Controller.cs`**: habilitación de la recopilación de consultas SQL para la instancia

```
using Amazon;
using Amazon.Util;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.SqlServer;

private void QuerySql(int id)
{
  var connectionString = ConfigurationManager.AppSettings["RDS_CONNECTION_STRING"];
  using (var sqlConnection = new SqlConnection(connectionString))
  using (var command = new TraceableSqlCommand("SELECT " + id, sqlConnection,
 collectSqlQueries: true))
  {
    command.ExecuteNonQuery();
  }
}
```

Creación de subsegmentos adicionales

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Los subsegmentos amplían el segmento de un rastro con detalles sobre el trabajo realizado para atender una solicitud. Cada vez que usted realiza una llamada con un cliente instrumentado, el SDK de X-Ray registra la información generada en un subsegmento. Puede crear subsegmentos adicionales para agrupar otros subsegmentos, medir el rendimiento de una sección de código o registrar anotaciones y metadatos.

Para administrar los subsegmentos, utilice los métodos `BeginSubsegment` y `EndSubsegment`. Realice el trabajo que desee en el subsegmento en un bloque `try` y utilice `AddException` para rastrear excepciones. Llame a `EndSubsegment` en un bloque `finally` para asegurarse de que el subsegmento está cerrado.

Example Controller.cs: subsegmento personalizado

```
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
try
{
  DoWork();
}
catch (Exception e)
{
  AWSXRayRecorder.Instance.AddException(e);
}
finally
{
  AWSXRayRecorder.Instance.EndSubsegment();
}
```

Cuando crea un subsegmento dentro de un segmento o de otro subsegmento, el SDK de X-Ray para .NET genera un ID para dicho subsegmento y registra la hora de inicio y la hora de finalización.

Example Subsegmentos con metadatos

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

Agregue anotaciones y metadatos a los segmentos con el SDK de X-Ray para .NET

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede registrar información adicional acerca de las solicitudes, el entorno o su aplicación con anotaciones y metadatos. Puede añadir anotaciones y metadatos a los segmentos que crea el SDK de X-Ray o a los subsegmentos personalizados que cree usted mismo.

Las anotaciones son pares de clave-valor con valores de cadena, numéricos o booleanos. Las anotaciones se indexan para su uso con expresiones de filtro. Utilice anotaciones para registrar los datos que desee utilizar para agrupar rastros en la consola o cuando llame a la API de `GetTraceSummaries`.

Los metadatos son pares de clave-valor con valores de cualquier tipo, por ejemplo objetos y listas, pero que no se indexan para utilizarlos con expresiones de filtro. Utilice los metadatos para registrar datos adicionales que desee almacenar en el rastro, pero que no necesite usar para hacer búsquedas.

Secciones

Registro de anotaciones con el SDK de X-Ray para .NET

Utilice anotaciones para registrar información sobre segmentos o subsegmentos que desee indexar para las búsquedas.

Se requiere lo siguiente para todas las anotaciones de X-Ray:

Requisitos de anotación

- Claves: la clave de una anotación de X-Ray puede contener hasta 500 caracteres alfanuméricos. No se pueden usar espacios ni símbolos, excepto el punto (.)
- Valores: el valor de una anotación de X-Ray puede contener hasta 1000 caracteres Unicode.
- Número de anotaciones: se pueden utilizar hasta 50 anotaciones por rastro.

Para grabar anotaciones fuera de una función AWS Lambda

1. Obtenga una instancia de `AWSXRayRecorder`.

   ```
   using Amazon.XRay.Recorder.Core;
   ...
   AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
   ```

2. Llame a `addAnnotation` con una clave de cadena y un valor booleano, Int32, Int64, doble o de cadena.

   ```
   recorder.AddAnnotation("mykey", "my value");
   ```

   El siguiente ejemplo muestra cómo llamar a `putAnnotation` con una clave de cadena que incluye un punto y un valor booleano, numérico o de cadena.

```
document.putAnnotation("testkey.test", "my value");
```

Para grabar anotaciones dentro de una función AWS Lambda

Tanto los segmentos como los subsegmentos de una función de Lambda se administran mediante el entorno de tiempo de ejecución de Lambda. Si desea añadir una anotación a un segmento o subsegmento dentro de una función de Lambda, debe hacer lo siguiente:

1. Cree el segmento o subsegmento dentro de la función de Lambda.
2. Añada la anotación al segmento o subsegmento.
3. Finalice el segmento o subsegmento.

En el siguiente código de ejemplo, se muestra cómo agregar una anotación a un subsegmento de una función de Lambda:

```
#Create the subsegment
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
#Add an annotation
AWSXRayRecorder.Instance.AddAnnotation("My", "Annotation");
try
{
  YourProcess(); #Your function
}
catch (Exception e)
{
  AWSXRayRecorder.Instance.AddException(e);
}
finally #End the subsegment
{
  AWSXRayRecorder.Instance.EndSubsegment();
}
```

El SDK de X-Ray registra las anotaciones como pares de clave-valor en un objeto `annotations` del documento de segmento. Si llama dos veces a una operación `addAnnotation` con la misma clave, se sobrescribe el valor previamente registrado en ese segmento o subsegmento.

Para encontrar rastros que tengan anotaciones con valores específicos, utilice la palabra clave `annotation[`*`key`*`]` en una expresión de filtro.

Registro de metadatos con el SDK de X-Ray para .NET

Utilice los metadatos para registrar información sobre segmentos o subsegmentos que no necesite indexar para emplearlos en las búsquedas. Los valores de metadatos pueden ser cadenas, números, booleanos o cualquier otro objeto que se pueda serializar en un objeto o matriz JSON.

Para registrar metadatos

1. Obtenga una instancia de `AWSXRayRecorder`, tal como se muestra en el siguiente código de ejemplo:

   ```
   using Amazon.XRay.Recorder.Core;
   ...
   AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
   ```

2. Llame a `AddMetadata` con un espacio de nombres de cadena, una clave de cadena y un valor de objeto, como se indica en el siguiente código de ejemplo:

   ```
   recorder.AddMetadata("my namespace", "my key", "my value");
   ```

   También puede llamar a la operación `AddMetadata` con solo un par de clave-valor, como se muestra en el siguiente código de ejemplo:

   ```
   recorder.AddMetadata("my key", "my value");
   ```

Si no especifica un valor para el espacio de nombres, el SDK de X-Ray utiliza `default`. Si llama dos veces a una operación `AddMetadata` con la misma clave, se sobrescribe el valor previamente registrado en ese segmento o subsegmento.

Uso de Ruby

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede instrumentar su aplicación Ruby de dos maneras para enviar rastros a X-Ray:

- AWS Distro for OpenTelemetry Ruby: una AWS distribución que proporciona un conjunto de bibliotecas de código abierto para enviar métricas y rastreos correlacionados a múltiples soluciones de AWS monitoreo, incluidas Amazon y Amazon OpenSearch Service CloudWatch AWS X-Ray, a través de AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK para Ruby: conjunto de bibliotecas para generar y enviar trazas a X-Ray mediante el daemon X-Ray.

Para obtener más información, consulte Cómo elegir entre AWS Distro for OpenTelemetry y X-Ray SDKs.

AWS Distro para OpenTelemetry Ruby

Con AWS Distro para OpenTelemetry (ADOT) Ruby, puede instrumentar sus aplicaciones una vez y enviar métricas y rastros correlacionados a varias soluciones de supervisión de AWS, incluidas Amazon CloudWatch, AWS X-Ray y Amazon OpenSearch Service. El uso de X-Ray con ADOT requiere dos componentes: un SDK de OpenTelemetry habilitado para su uso con X-Ray y AWS Distro para OpenTelemetry Collector habilitado para su uso con X-Ray.

Para empezar, consulte la documentación de AWS Distro para OpenTelemetry Ruby.

Para obtener más información sobre el uso de AWS Distro para OpenTelemetry con AWS X-Ray y otros Servicios de AWS, consulte AWS Distro para OpenTelemetry o la documentación de AWS Distro para OpenTelemetry.

Para obtener información adicional sobre los lenguajes compatibles y el uso, consulte AWS Observability en GitHub.

AWS X-Ray SDK for Ruby

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray es una biblioteca para las aplicaciones web de Ruby que proporciona clases y métodos para generar y enviar datos de rastreo al daemon de X-Ray. Los datos de rastreo incluyen información sobre las solicitudes HTTP entrantes atendidas por la aplicación y las llamadas que la aplicación realiza a los servicios descendentes mediante el AWS SDK, los clientes HTTP o un cliente de registro activo. También puede crear segmentos de forma manual y agregar información de depuración en anotaciones y metadatos.

Puede descargar el SDK añadiéndolo a su archivo Gemfile y ejecutando `bundle install`.

Example Archivo Gemfile

```
gem 'aws-sdk'
```

Si utiliza Rails, en primer lugar añada el middleware del SDK de X-Ray para rastrear las solicitudes entrantes. Un filtro de solicitudes crea un segmento. Mientras el segmento está abierto, puede utilizar los métodos del cliente del SDK para añadir información al segmento y crear subsegmentos para rastrear llamadas posteriores. El SDK también registra automáticamente las excepciones que

produce su aplicación mientras el segmento está abierto. Para las aplicaciones que no son de Rails, puede crear segmentos manualmente.

A continuación, utilice el SDK de X-Ray para AWS SDK for Ruby instrumentar sus clientes HTTP y SQL configurando la grabadora para que aplique parches a las bibliotecas asociadas. Cada vez que realizas una llamada a un recurso Servicio de AWS o a un canal intermedio con un cliente instrumentado, el SDK registra la información sobre la llamada en un subsegmento. Servicios de AWS y los recursos a los que accedes desde los servicios aparecen como nodos descendentes en el mapa de rastreo para ayudarte a identificar los errores y los problemas de limitación en las conexiones individuales.

En cuanto empiece a utilizar el SDK, personalice su comportamiento configurando la grabadora. Puede añadir complementos para registrar datos sobre los recursos informáticos que ejecutan la aplicación, personalizar el comportamiento de muestreo mediante la definición de reglas de muestreo y proporcionar un registrador para ver más o menos información del SDK en los logs de la aplicación.

Registre información adicional acerca de las solicitudes y el trabajo que la aplicación realiza en anotaciones y metadatos. Las anotaciones son pares sencillos de clave-valor que se indexan para su uso con expresiones de filtro para poder buscar rastros que contengan datos específicos. Las entradas de metadatos son menos restrictivas y pueden registrar objetos y matrices completos, es decir, todo lo que se pueda serializar en JSON.

> **Anotaciones y metadatos**
>
> Las anotaciones y los metadatos son texto arbitrario que se agrega a los segmentos con el SDK de X-Ray. Las anotaciones se indexan para su uso con expresiones de filtro. Los metadatos no se indexan pero se pueden ver en el segmento sin procesar con la consola o la API de X-Ray. Cualquier persona a la que conceda acceso de lectura a X-Ray puede ver estos datos.

Cuando tenga muchos clientes instrumentados en su código, un único segmento de solicitud puede contener un gran número de subsegmentos, uno para cada llamada realizada con un cliente instrumentado. Puede organizar y agrupar los subsegmentos incluyendo las llamadas del cliente en subsegmentos personalizados. Puede crear un subsegmento personalizado para una función completa o para cualquier sección de código, y registrar los metadatos y las anotaciones en el subsegmento en lugar de escribirlo todo en el segmento principal.

Para acceder a la documentación de referencia de las clases y los métodos del SDK, consulte la Referencia de la API del SDK de AWS X-Ray para Ruby.

Requisitos

El SDK de X-Ray requiere Ruby 2.3 o posterior y es compatible con las siguientes bibliotecas:

- AWS SDK for Ruby versión 3.0 o posterior
- Rails versión 5.1 o posterior

Configuración del SDK de X-Ray para Ruby

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

El SDK de X-Ray para Ruby tiene una clase denominada `XRay.recorder` que proporciona la grabadora global. Puede configurar la grabadora global para que personalice el middleware que crea los segmentos para las llamadas HTTP entrantes.

Secciones

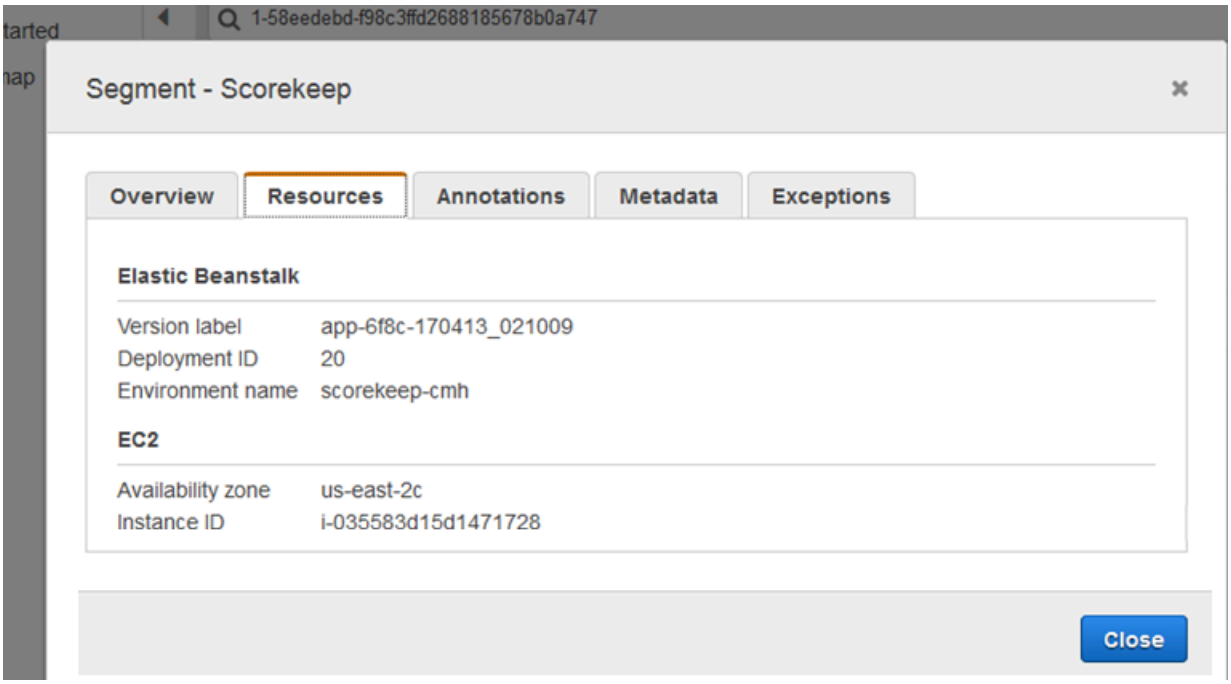
- Complementos del servicio
- Reglas de muestreo
- Registro
- Configuración de la grabadora en código
- Configuración de la grabadora con Rails
- Variables de entorno

Complementos del servicio

Utilice `plugins` para registrar información sobre el servicio que aloja la aplicación.

Plugins

- Amazon EC2 : `ec2` añade el ID de instancia y la zona de disponibilidad.
- Elastic Beanstalk: `elastic_beanstalk` añade el nombre de entorno, la etiqueta de versión y el ID de implementación.
- Amazon ECS: `ecs` agrega el ID de contenedor.



Para utilizar complementos, especifíquelos en el objeto de configuración que se pasa a la grabadora.

Example main.rb: configuración de complementos

```
my_plugins = %I[ec2 elastic_beanstalk]

config = {
  plugins: my_plugins,
  name: 'my app',
}

XRay.recorder.configure(config)
```

También puede utilizar las variables de entorno, que tienen prioridad frente a los valores establecidos en código, para configurar la grabadora.

El SDK también usa la configuración del complemento para establecer el campo `origin` en el segmento. Esto indica el tipo de AWS recurso que ejecuta la aplicación. Cuando utilizas varios complementos, el SDK utiliza el siguiente orden de resolución para determinar el origen: ElasticBeanstalk > EKS > ECS > EC2.

Reglas de muestreo

El SDK utiliza las reglas de muestreo que define el usuario en la consola de X-Ray para determinar qué solicitudes registrar. La regla predeterminada rastrea la primera solicitud cada segundo y el 5 % de las solicitudes adicionales de todos los servicios que envían rastros a X-Ray. Cree reglas adicionales en la consola de X-Ray para personalizar la cantidad de datos registrados para cada una de sus aplicaciones.

El SDK aplica las reglas personalizadas en el orden en que se definen. Si una solicitud coincide con varias reglas personalizadas, el SDK solo aplica la primera regla.

> **Note**
>
> Si el SDK no puede comunicarse con X-Ray para obtener las reglas de muestreo, vuelve a una regla local predeterminada de la primera solicitud cada segundo y del 5 % de las solicitudes adicionales por host. Esto puede ocurrir si el host no tiene permiso para llamar al muestreo APIs o no puede conectarse al daemon X-Ray, que actúa como proxy TCP para las llamadas a la API realizadas por el SDK.

También puede configurar el SDK para que cargue las reglas de muestreo desde un documento JSON. El SDK puede usar las reglas locales como respaldo para los casos en que el muestreo de X-Ray no esté disponible, o puede usar las reglas locales exclusivamente.

Example sampling-rules.json

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
```

```
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

En este ejemplo se define una regla personalizada y una regla predeterminada. La regla personalizada aplica un porcentaje de muestreo del 5 % sin un número mínimo de solicitudes de rastreo para las rutas incluidas bajo `/api/move/`. La regla predeterminada rastrea la primera solicitud cada segundo y el 10 % de las solicitudes adicionales.

La desventaja de definir las reglas localmente es que el objetivo establecido lo aplica cada instancia de la grabadora de forma independiente, en lugar de ser administrado por el servicio de X-Ray. A medida que se implementan más hosts, el porcentaje establecido se multiplica, lo que dificulta el control de la cantidad de datos registrados.

Para configurar reglas de copia de seguridad, defina un hash para el documento en el objeto de configuración que se pasa a la grabadora.

Example main.rb: configuración de la regla de copia de seguridad

```
require 'aws-xray-sdk'
my_sampling_rules =  {
  version: 1,
  default: {
    fixed_target: 1,
    rate: 0.1
  }
}
config = {
  sampling_rules: my_sampling_rules,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Para almacenar las reglas de muestreo por separado, defina el hash en un archivo independiente y haga que el archivo lo incorpore a la aplicación.

Example config/sampling-rules.rb

```
my_sampling_rules =  {
  version: 1,
  default: {
    fixed_target: 1,
    rate: 0.1
  }
}
```

Example main.rb: regla de muestreo desde un archivo

```
require 'aws-xray-sdk'
require 'config/sampling-rules.rb'

config = {
  sampling_rules: my_sampling_rules,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Para utilizar solo reglas locales, requiera las reglas de muestreo y configure el `LocalSampler`.

Example main.rb: muestreo con reglas locales

```
require 'aws-xray-sdk'
require 'aws-xray-sdk/sampling/local/sampler'

config = {
  sampler: LocalSampler.new,
  name: 'my app',
}
XRay.recorder.configure(config)
```

También puede configurar la grabadora global para deshabilitar el muestreo e instrumentar todas las solicitudes entrantes.

Example main.rb: inhabilitar el muestreo

```
require 'aws-xray-sdk'
config = {
  sampling: false,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Registro

De forma predeterminada, la grabadora muestra los eventos de nivel informativo en `$stdout`. Puede personalizar el registro mediante la definición de un registrador en el objeto de configuración que se pasa a la grabadora.

Example main.rb: registro

```
require 'aws-xray-sdk'
config = {
  logger: my_logger,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Utilice registros de depuración para identificar problemas como la presencia de subsegmentos sin cerrar al generar subsegmentos de forma manual.

Configuración de la grabadora en código

Hay disponibles ajustes adicionales a partir del método `configure` en `XRay.recorder`.

- `context_missing`: establezca esta opción en `LOG_ERROR` para evitar que se produzcan excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.
- `daemon_address`: establezca el host y el puerto del oyente del daemon de X-Ray.
- `name`: establezca un nombre de servicio que el SDK utiliza para los segmentos.
- `naming_pattern`: establezca un patrón de asignación de nombres de dominio para utilizar la nomenclatura dinámica.

- `plugins`: registre información acerca de los recursos de AWS de su aplicación con complementos.
- `sampling`: establezca esta opción en `false` para deshabilitar el muestreo.
- `sampling_rules`: establezca el hash que contiene las reglas de muestreo.

Example main.rb: deshabilite excepciones de falta de contexto

```
require 'aws-xray-sdk'
config = {
  context_missing: 'LOG_ERROR'
}

XRay.recorder.configure(config)
```

Configuración de la grabadora con Rails

Si utiliza el marco de trabajo Rails, puede configurar las opciones de la grabadora global en un archivo de Ruby situado en `app_root/initializers`. El SDK de X-Ray admite una clave de configuración adicional para su uso con Rails.

- `active_record`: establezca esta opción en `true` para registrar subsegmentos para las transacciones de la base de datos de grabación activa.

Configure los ajustes disponibles en un objeto de configuración denominado `Rails.application.config.xray`.

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {
  name: 'my app',
  patch: %I[net_http aws_sdk],
  active_record: true
}
```

Variables de entorno

Puede usar variables de entorno con el fin de configurar el SDK de X-Ray para Ruby. El SDK admite las siguientes variables:

- `AWS_XRAY_TRACING_NAME`: establezca un nombre de servicio que el SDK utiliza para los segmentos. Anula el nombre de servicio que se ha establecido en la estrategia de nomenclatura de segmento del filtro de servlet.
- `AWS_XRAY_DAEMON_ADDRESS`: establezca el host y el puerto del oyente del daemon de X-Ray. De forma predeterminada, el SDK envía los datos de rastreo a `127.0.0.1:2000`. Use esta variable si ha configurado el daemon para que escuche en un puerto diferente o si se ejecuta en un host diferente.
- `AWS_XRAY_CONTEXT_MISSING`: establezca esta opción en `RUNTIME_ERROR` para generar excepciones cuando el código instrumentado intente registrar datos sin que haya ningún segmento abierto.

  Valores válidos

  - `RUNTIME_ERROR`: lance una excepción de tiempo de ejecución.
  - `LOG_ERROR`: registre un error y continúe (predeterminado).
  - `IGNORE_ERROR`: ignore el error y continúe.

  Se pueden producir errores relativos a segmentos o subsegmentos inexistentes al intentar usar un cliente instrumentado en el código de inicio que se ejecuta cuando no hay ninguna solicitud abierta o en el código que inicia un nuevo subproceso.

Las variables de entorno anulan los valores establecidos en el código.

Rastreo de las solicitudes entrantes con el SDK de X-Ray para middleware de Ruby

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede usar el SDK de X-Ray para rastrear las solicitudes HTTP entrantes que su aplicación sirve en una EC2 instancia de Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Si utiliza Rails, use el middleware de Rails para instrumentar las solicitudes HTTP entrantes. Cuando añade el middleware a su aplicación y configura un nombre de segmento, el SDK de X-Ray para Ruby crea un segmento para cada solicitud muestreada. Cualquier segmento que se haya creado con instrumentación adicional se convierte en subsegmentos del segmento en nivel de la solicitud, el cual provee información sobre la solicitud HTTP y la respuesta. Esta información incluye el momento, el método y la disposición de la solicitud.

Cada segmento tiene un nombre que identifica la aplicación en el mapa de servicio. El nombre del segmento se puede asignar de forma estática o se puede configurar el SDK para que le asigne un nombre dinámico en función del encabezado del host de la solicitud entrante. La nomenclatura dinámica permite agrupar los rastros en función del nombre de dominio de la solicitud y aplicar un nombre predeterminado si el nombre no coincide con el patrón esperado (por ejemplo, si el encabezado del host está falsificado).

> **Solicitudes reenviadas**
>
> Si un equilibrador de carga u otro intermediario reenvía una solicitud a la aplicación, X-Ray toma la IP de cliente del encabezado `X-Forwarded-For` de la solicitud en lugar de tomar la IP de origen del paquete IP. La IP de cliente que se graba para una solicitud reenviada puede estar falsificada, por lo que no se debe confiar en ella.

Cuando se reenvía una solicitud, el SDK crea un campo adicional en el segmento para indicar que se realizó esta acción. Si el segmento contiene el campo `x_forwarded_for` configurado en `true`, el IP del cliente se obtiene a partir del encabezado `X-Forwarded-For` en la solicitud HTTP.

El middleware crea un segmento para cada solicitud entrante con un bloque `http` que contiene la siguiente información:

- Método HTTP: GET, POST, PUT, DELETE, etc.
- Dirección del cliente: la dirección IP del cliente que envió la solicitud.
- Código de respuesta: el código de respuesta HTTP para la solicitud finalizada.
- Intervalo: la hora de inicio (cuando se recibió la solicitud) y la hora de finalización (cuando se envió la respuesta).

- Agente del usuario: el `user-agent` de la solicitud.
- Longitud del contenido: la `content-length` de la respuesta.

Uso del middleware de Rails

Para utilizar el middleware, actualice el archivo Gemfile para que incluya el railtie necesario.

Example Archivo Gemfile: rails

```
gem 'aws-xray-sdk', require: ['aws-xray-sdk/facets/rails/railtie']
```

Para utilizar el middleware, también debe configurar la grabadora con un nombre que represente a la aplicación en el mapa de rastros.

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {
  name: 'my app'
}
```

Instrumentación manual del código

Si no utiliza Rails, cree los segmentos manualmente. Puede crear un segmento para cada solicitud entrante o crear segmentos alrededor de clientes HTTP o AWS SDK parcheados para proporcionar contexto a la grabadora y añadir subsegmentos.

```
# Start a segment
segment = XRay.recorder.begin_segment 'my_service'
# Start a subsegment
subsegment = XRay.recorder.begin_subsegment 'outbound_call', namespace: 'remote'

# Add metadata or annotation here if necessary
my_annotations = {
  k1: 'v1',
  k2: 1024
}
segment.annotations.update my_annotations

# Add metadata to default namespace
```

```
subsegment.metadata[:k1] = 'v1'

# Set user for the segment (subsegment is not supported)
segment.user = 'my_name'

# End segment/subsegment
XRay.recorder.end_subsegment
XRay.recorder.end_segment
```

Configuración de una estrategia de nomenclatura de segmentos

AWS X-Ray utiliza un nombre de servicio para identificar la aplicación y distinguirla del resto de aplicaciones, bases de datos, recursos externos APIs y otros AWS recursos que utiliza la aplicación. Cuando el SDK de X-Ray genera segmentos para las solicitudes entrantes, registra el nombre del servicio de la aplicación en el campo de nombre del segmento.

El SDK de X-Ray puede nombrar los segmentos utilizando el nombre de host en el encabezado de la solicitud HTTP. Sin embargo, este encabezado se puede falsificar, lo que podría provocar nodos inesperados en el mapa de servicio. Para evitar que el SDK nombre los segmentos de forma incorrecta debido a que las solicitudes tienen encabezados de host falsificados, debe especificar un nombre predeterminado para las solicitudes entrantes.

Si la aplicación atiende solicitudes de varios dominios, puede configurar el SDK para que utilice una estrategia de nomenclatura dinámica que refleje esto en los nombres de los segmentos. Una estrategia de nomenclatura dinámica permite al SDK usar el nombre de host para las solicitudes que coinciden con un patrón esperado y aplicar el nombre predeterminado a las solicitudes que no coincidan.

Por ejemplo, es posible que tenga una sola aplicación que atienda solicitudes a tres subdominios: `www.example.com`, `api.example.com` y `static.example.com`. Puede usar una estrategia de nomenclatura dinámica con el patrón `*.example.com` para identificar los segmentos de cada subdominio con un nombre diferente, lo que da como resultado tres nodos de servicio en el mapa de servicio. Si su aplicación recibe solicitudes con un nombre de host que no coincide con el patrón, verá un cuarto nodo en el mapa de servicio con el nombre alternativo que especifique.

Si desea utilizar el mismo nombre para todos los segmentos de solicitud, especifique el nombre de la aplicación cuando configure la grabadora, como se muestra en las secciones anteriores.

Una estrategia de nomenclatura dinámica define un patrón con el que deben coincidir los nombres de host y un nombre predeterminado que se utiliza si el nombre de host de la solicitud HTTP no

coincide con el patrón. Para asignar nombre a los segmentos de forma dinámica, especifique un patrón de nomenclatura en el hash config.

Example main.rb: nomenclatura dinámica

```
config = {
  naming_pattern: '*mydomain*',
  name: 'my app',
}

XRay.recorder.configure(config)
```

Puede utilizar "*" en el patrón para buscar coincidencia con cualquier cadena o "?" para buscar coincidencias con cualquier carácter único.

> **Note**
>
> Puede anular el nombre de servicio predeterminado que ha definido en el código mediante la `AWS_XRAY_TRACING_NAME`variable de entorno???.

Aplicación de parches a bibliotecas para instrumentar llamadas posteriores

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Para instrumentar llamadas posteriores, utilice el SDK de X-Ray para Ruby con el fin de aplicar parches a las bibliotecas que utiliza la aplicación. El SDK de X-Ray para Ruby puede aplicar parches a las siguientes bibliotecas.

Bibliotecas compatibles

- `net/http`: instrumente clientes HTTP.
- `aws-sdk`— Clientes de instrumentos. AWS SDK for Ruby

Al utilizar una biblioteca con parches, el SDK de X-Ray para Ruby crea un subsegmento para la llamada y registra información desde la solicitud y la respuesta. Debe haber un segmento disponible para que el SDK cree el subsegmento, ya sea desde el middleware del SDK o mediante una llamada a `XRay.recorder.begin_segment`.

Para aplicar parches a las bibliotecas, especifíquelos en el objeto de configuración que se pasa a la grabadora de X-Ray.

Example main.rb: aplique parches a las bibliotecas

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http aws_sdk]
}

XRay.recorder.configure(config)
```

Rastreo de llamadas al AWS SDK con el SDK de X-Ray para Ruby

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Cuando tu aplicación realiza llamadas Servicios de AWS para almacenar datos, escribir en una cola o enviar notificaciones, el X-Ray SDK for Ruby rastrea las llamadas en sentido descendente

en subsegmentos. El rastreo Servicios de AWS y los recursos a los que accede dentro de esos servicios (por ejemplo, un bucket de Amazon S3 o una cola de Amazon SQS) aparecen como nodos descendentes en el mapa de rastreo de la consola X-Ray.

El X-Ray SDK for Ruby instrumenta automáticamente todos los clientes AWS del SDK cuando se aplica un parche a la `aws-sdk` biblioteca. No puede instrumentar clientes individuales.

Para todos los servicios, puede ver el nombre de la API a la que se llama en la consola de X-Ray. Para un subconjunto de servicios, el SDK de X-Ray agrega información al segmento para proporcionar una mayor granularidad en el mapa de servicio.

Por ejemplo, cuando realiza una llamada con un cliente instrumentado de DynamoDB, el SDK agrega el nombre de tabla al segmento para las llamadas que se dirigen a una tabla. En la consola, cada tabla aparece como nodo independiente en el mapa de servicio, con un nodo genérico de DynamoDB para las llamadas que no se dirigen a una tabla.

Example Subsegmento para una llamada a DynamoDB con el fin de guardar un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Cuando accede a recursos designados, las llamadas a los siguientes servicios crean nodos adicionales en el mapa de servicio. Las llamadas que no están dirigidas a recursos concretos crean un nodo genérico en el servicio.

- Amazon DynamoDB: nombre de tabla

- Amazon Simple Storage Service: nombre de bucket y de clave
- Amazon Simple Queue Service: nombre de cola

Generación de subsegmentos personalizados con el SDK de X-Ray

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Los subsegmentos amplían el segmento de un rastro con detalles sobre el trabajo realizado para atender una solicitud. Cada vez que usted realiza una llamada con un cliente instrumentado, el SDK de X-Ray registra la información generada en un subsegmento. Puede crear subsegmentos adicionales para agrupar otros subsegmentos, medir el rendimiento de una sección de código o registrar anotaciones y metadatos.

Para administrar los subsegmentos, utilice los métodos `begin_subsegment` y `end_subsegment`.

```
subsegment = XRay.recorder.begin_subsegment name: 'annotations', namespace: 'remote'
my_annotations = { id: 12345 }
subsegment.annotations.update my_annotations
XRay.recorder.end_subsegment
```

Para crear un subsegmento para una función, encapsúlelo en una llamada a `XRay.recorder.capture`.

```
XRay.recorder.capture('name_for_subsegment') do |subsegment|
  resp = myfunc() # myfunc is your function
  subsegment.annotations.update k1: 'v1'
  resp
end
```

Cuando crea un subsegmento dentro de un segmento o de otro subsegmento, el SDK de X-Ray genera un ID para dicho subsegmento y registra la hora de inicio y la hora de finalización.

Example Subsegmentos con metadatos

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

Adición de anotaciones y metadatos a los segmentos con el SDK de X-Ray para Ruby

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support. Recomendamos migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Puede registrar información adicional acerca de las solicitudes, el entorno o su aplicación con anotaciones y metadatos. Puede añadir anotaciones y metadatos a los segmentos que crea el SDK de X-Ray o a los subsegmentos personalizados que cree usted mismo.

Las anotaciones son pares de clave-valor con valores de cadena, numéricos o booleanos. Las anotaciones se indexan para su uso con expresiones de filtro. Utilice anotaciones para registrar los datos que desee utilizar para agrupar rastros en la consola o cuando llame a la API de `GetTraceSummaries`.

Los metadatos son pares de clave-valor con valores de cualquier tipo, por ejemplo objetos y listas, pero que no se indexan para utilizarlos con expresiones de filtro. Utilice los metadatos para registrar datos adicionales que desee almacenar en el rastro, pero que no necesite usar para hacer búsquedas.

Además de anotaciones y metadatos, también puede registrar cadenas de ID de usuario en los segmentos. IDs Los usuarios se registran en un campo independiente en los segmentos y se indexan para usarlos en la búsqueda.

Secciones

- Registro de anotaciones con el SDK de X-Ray para Ruby
- Registro de metadatos con el SDK de X-Ray para Ruby
- Grabación del usuario IDs con el SDK de X-Ray para Ruby

Registro de anotaciones con el SDK de X-Ray para Ruby

Utilice anotaciones para registrar información sobre segmentos o subsegmentos que desee indexar para las búsquedas.

Requisitos de anotación

- Claves: la clave de una anotación de X-Ray puede contener hasta 500 caracteres alfanuméricos. No se pueden usar espacios ni símbolos, excepto el punto (.)
- Valores: el valor de una anotación de X-Ray puede contener hasta 1000 caracteres Unicode.
- Número de anotaciones: se pueden utilizar hasta 50 anotaciones por rastro.

Para registrar anotaciones

1. Obtenga una referencia al segmento o subsegmento actual desde `xray_recorder`.

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

o

```
require 'aws-xray-sdk'
...
```

```
document = XRay.recorder.current_subsegment
```

2. Llame a `update` con un valor hash.

```
my_annotations = { id: 12345 }
document.annotations.update my_annotations
```

El siguiente es un ejemplo que muestra cómo llamar a `update` con una clave de anotación que contiene un punto.

```
my_annotations = { testkey.test: 12345 }
document.annotations.update my_annotations
```

El SDK registra las anotaciones como pares de clave-valor en un objeto `annotations` del documento de segmento. Si llama dos veces a `add_annotations` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Para encontrar rastros que tengan anotaciones con valores específicos, utilice la palabra clave `annotation[key]` en una expresión de filtro.

Registro de metadatos con el SDK de X-Ray para Ruby

Utilice los metadatos para registrar información sobre segmentos o subsegmentos que no necesite indexar para las búsquedas. Los valores de metadatos pueden ser cadenas, números, booleanos o cualquier objeto que se pueda serializar en un objeto o matriz JSON.

Para registrar metadatos

1. Obtenga una referencia al segmento o subsegmento actual desde `xray_recorder`.

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

o

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_subsegment
```

2. Llame a `metadata` con una clave de cadena; un valor booleano, numérico, de cadena o de objeto y un espacio de nombres de cadena.

```
my_metadata = {
  my_namespace: {
    key: 'value'
  }
}
subsegment.metadata my_metadata
```

Si llama dos veces a `metadata` con la misma clave, se sobrescriben los valores previamente registrados en ese segmento o subsegmento.

Grabación del usuario IDs con el SDK de X-Ray para Ruby

Registre IDs el usuario en los segmentos de solicitud para identificar al usuario que envió la solicitud.

Para registrar al usuario IDs

1. Obtenga una referencia al segmento actual desde `xray_recorder`.

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

2. Especifique el ID de cadena del usuario que envió la solicitud en el campo de usuario del segmento.

```
segment.user = 'U12345'
```

Puede establecer el usuario en los controladores para registrar el ID de usuario tan pronto como la aplicación comience a procesar la solicitud.

Para buscar rastros de un ID de usuario, utilice la palabra clave `user` en una expresión de filtro.

Cronología de X-Ray SDK y Daemon Support

La siguiente tabla muestra las fechas y el nivel de soporte para X-Ray SDKs y Daemon.

Fase de SDK y daemon	Fecha de inicio	Fecha de finalización	Soporte proporcionado
Disponibilidad general	N/D	25 de febrero de 2026	X-Ray SDKs y Daemon son totalmente compatibles. AWS proporciona versiones periódicas de SDK y daemon que incluyen correcciones de errores y de seguridad.
Modo de mantenimiento	25 de febrero de 2026	N/A	AWS limitará las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. No SDKs/ Daemon recibirán nuevas mejoras en sus funciones.

Le recomendamos que migre a OpenTelemetry soluciones para instrumentar su aplicación y enviar trazas a X-Ray AWS . Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry

Migración de la instrumentación de rayos X a la instrumentación OpenTelemetry

> **Note**
>
> Aviso de SDK/Daemon mantenimiento de X-Ray: el 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente los problemas de seguridad. Para obtener más información sobre la cronología del soporte, consulte Cronología de X-Ray SDK y Daemon Support.

X-Ray está pasando a ser OpenTelemetry (OTel) su principal estándar de instrumentación para el rastreo y la observabilidad de las aplicaciones. Este cambio estratégico se alinea AWS con las mejores prácticas de la industria y ofrece a los clientes una solución más completa, flexible y preparada para el futuro para sus necesidades de observabilidad. OpenTelemetrySu amplia adopción en la industria permite rastrear las solicitudes en diversos sistemas, incluidos aquellos externos AWS que pueden no integrarse directamente con X-Ray.

En este capítulo se ofrecen recomendaciones para una transición fluida y se hace hincapié en la importancia de migrar a soluciones OpenTelemetry basadas en sistemas para garantizar el soporte y el acceso continuos a las funciones más recientes en materia de observabilidad e instrumentación de aplicaciones.

Se recomienda adoptarla OpenTelemetry como solución de observabilidad para instrumentar su aplicación.

Temas

- Migre a Node.js OpenTelemetry
- Migre a.NET OpenTelemetry
- Migrar a OpenTelemetry Python
- Migre a Ruby OpenTelemetry

Comprensión OpenTelemetry

OpenTelemetry es un marco de observabilidad estándar del sector que proporciona protocolos y herramientas estandarizados para recopilar datos de telemetría. Ofrece un enfoque unificado para instrumentar, generar, recopilar y exportar datos de telemetría como métricas, registros y rastros.

Al migrar de X-Ray SDKs a OpenTelemetry, obtiene los siguientes beneficios:

- Soporte mejorado con el marco y la instrumentación de bibliotecas
- Soporte para lenguajes de programación adicionales
- Capacidades de instrumentación automática
- Opciones de configuración de muestreo flexibles
- Recopilación unificada de métricas, registros y rastros

El OpenTelemetry recopilador ofrece más opciones para los formatos de recopilación de datos y los destinos de exportación que el daemon X-Ray.

OpenTelemetry soporte en AWS

AWS proporciona múltiples soluciones para trabajar con OpenTelemetry:

- AWS Distro para OpenTelemetry

  Exporte OpenTelemetry trazas como segmentos a X-Ray.

  Para obtener más información, consulte AWS Distro for OpenTelemetry.
- CloudWatch Señales de aplicación

  Exporte OpenTelemetry trazas y métricas personalizadas para supervisar el estado de las aplicaciones.

  Para obtener más información, consulte Trabajo con señales de aplicaciones.

- CloudWatch OTel Punto final

  Exporte OpenTelemetry las trazas a X-Ray mediante el OTel punto final HTTP con OpenTelemetry instrumentación nativa.

  Para obtener más información, consulte Uso de puntos OTel finales.

Utilización con OpenTelemetry AWS CloudWatch

AWS CloudWatch admite los OpenTelemetry rastreos a través de la instrumentación de aplicaciones del lado del cliente y de AWS CloudWatch servicios nativos, como Application Signals, Trace, Map, Metrics y Logs. Para obtener más información, consulte OpenTelemetry.

Comprensión OpenTelemetry de los conceptos de migración

La siguiente tabla mapea los conceptos de X-Ray con sus OpenTelemetry equivalentes. La comprensión de estos mapeos le ayuda a traducir su instrumentación de rayos X existente a: OpenTelemetry

Concepto de X-Ray	OpenTelemetry concepto
Grabadora de X-Ray	Proveedor de rastreador y rastreadores
Complementos del servicio	Detectores de recursos
Segmento	(Servidor) Intervalo
Subsegmento	Intervalo (sin servidor)
Reglas de muestreo de X-Ray	OpenTelemetry Muestreo (personalizable)
Emisor de X-Ray	Exportador de intervalos (personalizable)
Anotaciones/metadatos	Atributos
Instrumentación de biblioteca	Instrumentación de biblioteca
Contexto de rastro de X-Ray	Contexto del intervalo
Propagación de contexto del rastro de X-Ray	Propagación de contexto de rastro de W3C

Concepto de X-Ray	OpenTelemetry concepto
Muestreo del rastro de X-Ray	OpenTelemetry Muestreo de rastros
N/A	Procesamiento del intervalo
N/A	Equipaje
Daemon de X-Ray	OpenTelemetry Coleccionista

> **Note**
>
> Para obtener más información sobre OpenTelemetry los conceptos, consulte la OpenTelemetry documentación.

Comparación de características

En la tabla siguiente se muestran las características admitidas en ambos servicios. Utilice esta información para identificar las brechas que necesite abordar durante la migración:

Característica	Instrumentación de X-Ray	OpenTelemetry instrumentación
Instrumentación de biblioteca	Soportado	compatible
Muestreo de X-Ray	compatible	Compatible con java/.net/GO OTel Compatible con ADOT Java/.NET/Python/Node.js
Propagación de contexto del rastro de X-Ray	Soportado	compatible
Detección de recursos	Soportado	compatible
Anotaciones de segmentos	Soportado	compatible

Característica	Instrumentación de X-Ray	OpenTelemetry instrumentación
Metadatos del segmento	Soportado	compatible
Instrumentación automática sin código	Compatible con Java	Compatible con OTel Java/.NET/Python/Node.js Compatible con ADOT Java/.NET/Python/Node.js
Creación de rastros manuales	Soportado	compatible

Instalación y configuración del rastro

Para crear trazas OpenTelemetry, necesitas un rastreador. Para obtener un rastreador, debe inicializar un proveedor de rastreadores en la aplicación. Esto es similar a la forma en que se utiliza la grabadora de X-Ray para configurar X-Ray y crear segmentos y subsegmentos en un rastro de X-Ray.

> **Note**
>
> El OpenTelemetry Tracer Provider ofrece más opciones de configuración que el grabador de rayos X.

Descripción de la estructura de datos de rastros

Tras comprender los conceptos básicos y las asignaciones de características, podrá obtener información sobre los detalles específicos de la implementación, como la estructura de los datos de rastros y el muestreo.

OpenTelemetry utiliza tramos en lugar de segmentos y subsegmentos para estructurar los datos de rastreo. Cada intervalo incluye los siguientes componentes:

- Name
- ID único
- Marcas de tiempo de inicio y finalización

- Tipo de intervalo
- Contexto del intervalo
- Atributos (metadatos clave-valor)
- Eventos (registros con fecha y hora)
- Enlaces a otros intervalos
- Información del estado
- Referencias de intervalo principal

Al migrar a OpenTelemetry, los tramos se convierten automáticamente en segmentos o subsegmentos de X-Ray. Esto garantiza que su experiencia de CloudWatch consola actual permanezca inalterada.

Uso de atributos de intervalo

El X-Ray SDK ofrece dos formas de agregar datos a segmentos y subsegmentos:

Anotaciones

Pares clave-valor que se indexan para filtrar y buscar

Metadatos

Pares clave-valor que contienen datos complejos que no están indexados para la búsqueda

De forma predeterminada, los OpenTelemetry atributos de espacio se convierten en metadatos en los datos sin procesar de X-Ray. Para convertir atributos específicos en anotaciones, agregue sus claves a la lista de atributos `aws.xray.annotations`.

- Para obtener más información sobre OpenTelemetry los conceptos, consulte OpenTelemetry Traces
- Para obtener más información sobre cómo OpenTelemetry los datos se asignan a los datos de rayos X, consulte OpenTelemetry Traducción de modelos de datos de rayos X

Detección de recursos en el entorno

OpenTelemetry usa Resource Detectors para recopilar metadatos sobre los recursos que generan datos de telemetría. Estos metadatos se almacenan como atributos de recurso. Por ejemplo, una entidad que produce telemetría podría ser un clúster de Amazon ECS o una EC2 instancia de

Amazon, y los atributos de recursos que se pueden registrar desde estas entidades pueden incluir el ARN del clúster de Amazon ECS o el ID de instancia de Amazon EC2 .

- Para obtener información sobre los tipos de recursos compatibles, consulte OpenTelemetry Convenciones semánticas de recursos
- Para obtener información acerca de los complementos del servicio de X-Ray, consulte Configuración de X-Ray SDK

Administración de las estrategias de muestreo

El muestreo de rastros le ayuda a administrar los costos al recopilar datos de un subconjunto representativo de solicitudes en lugar de todas las solicitudes. OpenTelemetry Tanto X-Ray como X-Ray admiten el muestreo, pero lo implementan de manera diferente.

> **Note**
>
> El muestreo de menos del 100 % de los rastros reduce los costos de observabilidad y, al mismo tiempo, mantiene una visión significativa del rendimiento de la aplicación.

OpenTelemetry proporciona varias estrategias de muestreo integradas y le permite crear estrategias personalizadas. También puede configurar un muestreador remoto de rayos X en algunos lenguajes del SDK para utilizar las reglas de muestreo de rayos X. OpenTelemetry

Las estrategias de muestreo adicionales OpenTelemetry son las siguientes:

- Muestreo basado en el principal: respeta la decisión de muestreo del intervalo principal antes de aplicar estrategias de muestreo adicionales
- Muestreo basado en la proporción de rastros: >Muestrea aleatoriamente un porcentaje específico de intervalos
- Muestreo de cola: aplica las reglas de muestreo para completar las trazas en el OpenTelemetry colector
- Muestreadores personalizados: implemente su propia lógica de muestreo mediante la interfaz de muestreo

Para obtener información sobre las reglas de muestreo de X-Ray, consulte Reglas de muestreo en la consola de X-Ray

Para obtener información sobre el muestreo de OpenTelemetry la cola, consulte el procesador de muestreo de la cola

Administración del contexto de rastros

X-Ray SDKs gestiona el contexto del segmento para gestionar correctamente las relaciones padre-hijo entre los segmentos y subsegmentos de una traza. OpenTelemetry utiliza un mecanismo similar para garantizar que los tramos tengan el intervalo principal correcto. Almacena y propaga los datos de rastreo en todo el contexto de una solicitud. Por ejemplo, cuando la aplicación procese una solicitud y cree un intervalo de servidor que represente esa solicitud, OpenTelemetry almacenará el intervalo de servidores en el OpenTelemetry contexto para que, cuando se cree un intervalo secundario, ese intervalo secundario pueda hacer referencia al intervalo del contexto como su principal.

Propagación del contexto de rastro

Tanto X-Ray como yo OpenTelemetry utilizan encabezados HTTP para propagar el contexto de rastreo en todos los servicios. Esto le permite vincular los datos de rastros generados por diferentes servicios y mantener las decisiones de muestreo.

El X-Ray SDK propaga automáticamente el contexto de rastro mediante el encabezado de rastro de X-Ray. Cuando un servicio llama a otro, el encabezado de rastro contiene el contexto necesario para mantener las relaciones principal-secundario entre los rastros.

OpenTelemetry admite varios formatos de encabezados de rastreo para la propagación del contexto, entre los que se incluyen:

- Contexto de rastro de W3C (predeterminado)
- Encabezado de rastro de X-Ray
- Otros formatos personalizados

> **Note**
>
> Puede configurarlo OpenTelemetry para usar uno o más formatos de encabezado. Por ejemplo, utilice el propagador de rayos X para enviar el contexto de rastreo a AWS los servicios que admiten el rastreo de rayos X.

Configure y utilice el propagador de rayos X para permitir el rastreo entre AWS los servicios. Esto le permite propagar el contexto de rastros a los puntos de conexión de API Gateway y otros servicios compatibles con X-Ray.

- Para obtener información acerca de los encabezados de rastro de X-Ray, consulte Encabezado de rastreo en la Guía para desarrolladores de X-Ray
- Para obtener información sobre la propagación OpenTelemetry del contexto, consulte Contexto y propagación del contexto en la documentación OpenTelemetry

Uso de la instrumentación de biblioteca

Tanto X-Ray como OpenTelemetry yo proporcionan instrumentación de biblioteca que requiere cambios mínimos en el código para añadir el rastreo a sus aplicaciones.

X-Ray proporciona funcionalidades de instrumentación de biblioteca. Esto le permite agregar instrumentaciones de X-Ray prediseñadas con cambios mínimos en el código de la aplicación. Estas instrumentaciones son compatibles con bibliotecas específicas, como el AWS SDK y los clientes HTTP, así como con marcos web como Spring Boot o Express.js.

OpenTelemetryLas bibliotecas de instrumentación generan extensiones detalladas para sus bibliotecas mediante enlaces a las bibliotecas o mediante la modificación automática del código, lo que requiere cambios mínimos en el código.

Para determinar si Library Instrumentations OpenTelemetry es compatible con su biblioteca, búsquela en el OpenTelemetry Registro de Registry. OpenTelemetry

Exportación de rastros

X-Ray y OpenTelemetry utilizan diferentes métodos para exportar los datos de rastreo.

Exportación de rastros de X-Ray

Los X-Ray SDKs utilizan un emisor para enviar datos de rastreo:

- Envía segmentos y subsegmentos al X-Ray Daemon
- Utiliza UDP para E/S sin bloqueo
- Configurado de forma predeterminada en el SDK

OpenTelemetry exportación de trazas

OpenTelemetry utiliza Span Exporters configurables para enviar datos de rastreo:

- Usa los protocolos http/protobuf o grpc
- Exporta los tramos a los puntos finales supervisados por el OpenTelemetry recopilador o el agente CloudWatch
- Permite configurar los exportadores de forma personalizada

Procesamiento y reenvío de rastros

Tanto X-Ray como OpenTelemetry yo proporcionan componentes para recibir, procesar y reenviar datos de rastreo.

Procesamiento de rastros de X-Ray

El X-Ray Daemon gestiona el procesamiento de rastros:

- Escucha el tráfico UDP de X-Ray SDKs
- Segmentos y subsegmentos por lotes
- Carga lotes al servicio de X-Ray

OpenTelemetry procesamiento de trazas

El OpenTelemetry recopilador gestiona el procesamiento de trazas:

- Recibe rastros de servicios instrumentados
- Procesa y, opcionalmente, modifica los datos de rastros
- Envía los rastros procesados a varios backends, incluido X-Ray

> **Note**
>
> El AWS CloudWatch agente también puede recibir y enviar OpenTelemetry trazas a X-Ray. Para obtener más información, consulte Recopile métricas y rastreos con OpenTelemetry.

Procesamiento de intervalos (concepto OpenTelemetry específico)

OpenTelemetry utiliza los procesadores Span para modificar los intervalos a medida que se crean:

- Permite leer y modificar los intervalos al crearlos o completarlos
- Permite una lógica personalizada para el manejo de los intervalos

Equipaje (OpenTelemetryconcepto específico)

OpenTelemetryLa función de equipaje permite la propagación de datos clave-valor:

- Permite pasar datos arbitrarios junto con el contexto de rastro
- Útil para propagar información específica de la aplicación más allá de los límites del servicio

Para obtener información sobre el OpenTelemetry recopilador, consulte el recopilador OpenTelemetry

Para obtener información acerca de los conceptos de X-Ray, consulte Conceptos de X-Ray en la Guía para desarrolladores de X-Ray

Información general sobre la migración

En esta sección, se proporciona información general acerca de los cambios de código necesarios para la migración. La siguiente lista contiene una guía específica para cada idioma y los pasos de migración de X-Ray Daemon.

> **Important**
>
> Para migrar completamente de la instrumentación de rayos X a OpenTelemetry la instrumentación, necesita:
>
> 1. Reemplace el uso del SDK de X-Ray por una OpenTelemetry solución
> 2. Sustituya el X-Ray Daemon por el CloudWatch agente o el OpenTelemetry colector (con X-Ray Exporter)

- Migración a Java OpenTelemetry
- Migre a OpenTelemetry Go

- Migre a Node.js OpenTelemetry
- Migre a.NET OpenTelemetry
- Migrar a OpenTelemetry Python
- Migre a Ruby OpenTelemetry

Recomendaciones para aplicaciones nuevas y existentes

Para las aplicaciones nuevas y existentes, se recomienda utilizar las siguientes soluciones para habilitar el rastreo en las aplicaciones:

Instrumentación

- OpenTelemetry SDKs
- AWS Distribución para instrumentación OpenTelemetry

Recopilación de datos

- OpenTelemetry Coleccionista
- CloudWatch Agente

Tras migrar a soluciones OpenTelemetry basadas en bases, su CloudWatch experiencia seguirá siendo la misma. Podrá seguir viendo sus trazas en el mismo formato en las páginas de trazas y mapas de trazas de la CloudWatch consola, o recuperar sus datos de rastreo a través del X-Ray APIs.

Rastreo de los cambios de configuración

Debe reemplazar la configuración de X-Ray por una OpenTelemetry configuración.

Comparación de X-Ray y OpenTelemetry configuración

Característica	X-Ray SDK	OpenTelemetry
Configuraciones predeterminadas	• Muestreo centralizado de X-Ray • Propagación de contexto del rastro de X-Ray • Exportación de rastros a X-Ray Daemon	• Exportación de trazas a un OpenTelemetry recopilador o CloudWatch agente (HTTP/gRPC) • Propagación de contexto de rastro de W3C

Característica	X-Ray SDK	OpenTelemetry
Configuraciones manuales	• Reglas de muestreo locales • Complementos de detección de recursos	• Muestreo de X-Ray (es posible que no esté disponible en todos los idiomas) • Detección de recursos • Propagación de contexto del rastro de X-Ray

Cambios de instrumentación de bibliotecas

Actualice su código para usar OpenTelemetry Library Instrumentation en lugar de X-Ray Library Instrumentation para AWS SDK, clientes HTTP, marcos web y otras bibliotecas. Esto genera OpenTelemetry trazas en lugar de trazas de X-Ray.

> **Note**
>
> Los cambios en el código varían según el idioma y la biblioteca. Consulte las guías de migración de idiomas específicos para obtener instrucciones detalladas.

Cambios en la instrumentación del entorno de Lambda

Para utilizarlas OpenTelemetry en las funciones de Lambda, elija una de estas opciones de configuración:

1. Utilice una capa de Lambda de instrumentación automática:
   - Capa Lambda de señales de CloudWatch aplicación (recomendada)

     > **Note**
     >
     > Para utilizar solo el rastreo, defina la variable de entorno de Lambda `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`.

   - AWS Capa Lambda gestionada para ADOT
2. Configuración manual OpenTelemetry para la función Lambda:

- Configuración de un procesador de intervalos simple con un exportador de intervalos de X-Ray UDP
- Configuración de un propagador de X-Ray Lambda

Creación manual de datos de rastros

Sustituya los segmentos y subsegmentos de X-Ray por OpenTelemetry Spans:

- Usa un OpenTelemetry rastreador para crear tramos
- Agregación de atributos a intervalos (equivalentes a anotaciones y metadatos de X-Ray)

> **Important**
>
> Cuando se envía a X-Ray:
>
> - Los intervalos de servidor se convierten en segmentos de X-Ray
> - Otros intervalos se convierten en subsegmentos de X-Ray
> - Los atributos se convierten en metadatos de forma predeterminada

Para convertir un atributo en una anotación, agregue su clave a la lista de atributos de `aws.xray.annotations`. Para obtener más información, consulte Habilitar anotaciones de X-Ray personalizadas.

Migración de X-Ray Daemon a AWS CloudWatch agente o recopilador OpenTelemetry

Puede utilizar el CloudWatch agente o el OpenTelemetry recopilador para recibir las trazas de sus aplicaciones instrumentadas y enviarlas a X-Ray.

> **Note**
>
> La versión 1.300025.0 del CloudWatch agente y las versiones posteriores pueden recopilar trazas. OpenTelemetry El uso del CloudWatch agente en lugar del Demonio X-Ray reduce

la cantidad de agentes que hay que gestionar. Para obtener más información, consulte Recopilación de métricas, registros y seguimientos con el CloudWatch agente.

Secciones

Migración en Amazon EC2 o en servidores locales

Important

Detenga el proceso de X-Ray Daemon antes de utilizar el CloudWatch agente o el OpenTelemetry recopilador para evitar conflictos en los puertos.

Configuración del X-Ray Daemon existente

Instalación del daemon

Su uso actual de X-Ray Daemon se instaló mediante uno de estos métodos:

Instalación manual

Descargue y ejecute el archivo ejecutable desde el bucket de Amazon S3 de X-Ray daemon.

Instalación automática

Utilice este script para instalar el daemon al lanzar una instancia:

```
#!/bin/bash
curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm \
    -o /home/ec2-user/xray.rpm
yum install -y /home/ec2-user/xray.rpm
```

Configuración del daemon de

Su uso actual de X-Ray Daemon se configuró mediante:

- Argumentos de línea de comandos
- Archivo de configuración () `xray-daemon.yaml`

Example Uso de un archivo de configuración

```
./xray -c ~/xray-daemon.yaml
```

Ejecutar el demonio

Su uso actual de X-Ray Daemon se inició con el siguiente comando:

```
~/xray-daemon$ ./xray -o -n us-east-1
```

Eliminación del daemon

Para eliminar el Demonio X-Ray de tu EC2 instancia de Amazon:

1. Detenga el servicio de daemon:

   ```
   systemctl stop xray
   ```

2. Elimine el archivo de configuración:

   ```
   rm ~/path/to/xray-daemon.yaml
   ```

3. Si está configurado, elimine el archivo de registro:

   > **Note**
   >
   > La ubicación del archivo de registro depende de la configuración:
   >
   > - Configuración de línea de comandos: `/var/log/xray-daemon.log`
   > - Archivo de configuración: compruebe la configuración de `LogPath`

Configuración del agente CloudWatch

Instalación del agente

Para obtener instrucciones de instalación, consulte Instalación del CloudWatch agente en un servidor local.

Configuración del agente

1. Cree un archivo de configuración para habilitar la recopilación de rastros. Para obtener más información, consulte Crear el archivo de configuración del CloudWatch agente.
2. Configure permisos de IAM:
   - Adjunte un rol de IAM o especifique las credenciales del agente. Para obtener más información, consulte Configuración de roles de IAM.
   - Asegúrese de que el rol o las credenciales incluyan el permiso `xray:PutTraceSegments`.

Cómo iniciar el agente de

Para obtener instrucciones sobre cómo iniciar el agente, consulte Iniciar el CloudWatch agente mediante la línea de comandos.

Configuración del OpenTelemetry recopilador

Instalación del recopilador

Descargue e instale el OpenTelemetry recopilador para su sistema operativo. Para obtener instrucciones, consulte Instalación del recopilador.

Configuración del recopilador

Configure los siguientes componentes en el recopilador:

- extensión awsproxy

  Necesario para el muestreo de X-Ray
- OTel receptores

  Recopila los rastros de la aplicación
- exportador de xray

Envía rastros a X-Ray

Example Ejemplo de configuración del recopilador: .yaml otel-collector-config

```
extensions:
  awsproxy:
    endpoint: 127.0.0.1:2000
  health_check:

receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 127.0.0.1:4317
      http:
        endpoint: 127.0.0.1:4318

processors:
  batch:

exporters:
  awsxray:
    region: 'us-east-1'

service:
  pipelines:
    traces:
      receivers: [otlp]
      exporters: [awsxray]
  extensions: [awsproxy, health_check]
```

Important

Configure AWS las credenciales con el `xray:PutTraceSegments` permiso. Para obtener más información, consulte Especificar credenciales.

Inicio del recopilador

Ejecute el recopilador con el archivo de configuración:

```
otelcol --config=otel-collector-config.yaml
```

Migración en Amazon ECS

> **Important**
>
> El rol de la tarea debe tener el permiso `xray:PutTraceSegments` de todos los recopiladores que utilice.
> Detenga cualquier contenedor de X-Ray Daemon existente antes de ejecutar el contenedor del CloudWatch agente o del OpenTelemetry recopilador en el mismo host para evitar conflictos de puertos.

Uso del agente CloudWatch

1. Obtenga la imagen de Docker de la Galería pública de Amazon ECR.
2. Cree un archivo de configuración denominado`cw-agent-otel.json`:

```
{
  "traces": {
    "traces_collected": {
      "xray": {
        "tcp_proxy": {
          "bind_address": "0.0.0.0:2000"
        }
      },
      "otlp": {
        "grpc_endpoint": "0.0.0.0:4317",
        "http_endpoint": "0.0.0.0:4318"
      }
    }
  }
}
```

3. Almacene la configuración en el almacén de parámetros de Systems Manager:
    1. Abra el icono https://console.aws.amazon.com/systems-manager/
    2. Elija Crear parámetro
    3. Escriba los siguientes valores:

- Nombre: `/ecs/cwagent/otel-config`
- Nivel: estándar
- Tipo: cadena
- Tipo de datos: texto
- Valor: [Pegue aquí la configuración cw-agent-otel .json]

4. Cree una definición de tarea mediante el modo de red puente:

   En la definición de tarea, la configuración depende del modo de red que se utilice. El modo de red puente es la opción predeterminada y se puede utilizar en la VPC predeterminada. En una red puente, defina la variable de `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` entorno para que indique al OpenTelemetry SDK cuál es el punto final y el puerto del CloudWatch agente. También debes crear un enlace desde el contenedor de tu aplicación al contenedor de Collector para que los seguimientos se envíen desde el OpenTelemetry SDK de tu aplicación al contenedor de Collector.

   Example CloudWatch definición de la tarea del agente

```
{
    "containerDefinitions": [
        {
            "name": "cwagent",
            "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
            "portMappings": [
                {
                    "containerPort": 4318,
                    "hostPort": 4318,
                    "protocol": "tcp"
                },
                {
                    "containerPort": 4317,
                    "hostPort": 4317,
                    "protocol": "tcp"
                },
                {
                    "containerPort": 2000,
                    "hostPort": 2000,
                    "protocol": "tcp"
                }
            ],
            "secrets": [
```

```
                {
                    "name": "CW_CONFIG_CONTENT",
                    "valueFrom": "/ecs/cwagent/otel-config"
                }
            ]
        },
        {
            "name": "application",
            "image": "APPLICATION_IMAGE",
            "links": ["cwagent"],
            "environment": [
                {
                    "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
                    "value": "http://cwagent:4318/v1/traces"
                }
            ]
        }
    ]
}
```

Para obtener más información, consulte Implementación del CloudWatch agente para recopilar métricas a EC2 nivel de instancia de Amazon en Amazon ECS.

Uso del recopilador OpenTelemetry

1. Obtenga la imagen de Docker `otel/opentelemetry-collector-contrib` de Docker Hub.
2. Cree un archivo de configuración denominado `otel-collector-config.yaml` con el mismo contenido que se muestra en la sección EC2 Configuración del recopilador de Amazon, pero actualice los puntos de enlace para usarlos `0.0.0.0` en lugar de`127.0.0.1`.
3. Para utilizar esta configuración en Amazon ECS, puede almacenar la configuración en el almacén de parámetros de Systems Manager. En primer lugar, vaya a la consola del almacén de parámetros de Systems Manager y elija Crear nuevo parámetro. Cree un nuevo parámetro con la siguiente información:
   - Nombre:/ecs/otel/config(se hará referencia a este nombre en la definición de tareas del recopilador)
   - Nivel: estándar
   - Tipo: cadena
   - Tipo de datos: texto

- Valor: [Pegue aquí la configuración otel-collector-config .yaml]

4. Cree una definición de tarea para implementar el OpenTelemetry recopilador utilizando el modo de red puente como ejemplo.

   En la definición de tarea, la configuración depende del modo de red que se utilice. El modo de red puente es la opción predeterminada y se puede utilizar en la VPC predeterminada. En una red puente, defina la variable de `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` entorno para indicar al OpenTelemetry SDK cuáles son el punto final y el puerto del OpenTelemetry recopilador. También debe crear un enlace desde el contenedor de la aplicación al contenedor del recopilador para que las trazas se envíen desde el OpenTelemetry SDK de la aplicación al contenedor del recopilador.

   Example OpenTelemetry definición de tareas de recopilación

```
{
    "containerDefinitions": [
        {
            "name": "otel-collector",
            "image": "otel/opentelemetry-collector-contrib",
            "portMappings": [
                {
                    "containerPort": 2000,
                    "hostPort": 2000
                },
                {
                    "containerPort": 4317,
                    "hostPort": 4317
                },
                {
                    "containerPort": 4318,
                    "hostPort": 4318
                }
            ],
            "command": [
                "--config",
                "env:SSM_CONFIG"
            ],
            "secrets": [
                {
                    "name": "SSM_CONFIG",
                    "valueFrom": "/ecs/otel/config"
                }
            ]
```

```
        },
        {
            "name": "application",
            "image": "APPLICATION_IMAGE",
            "links": ["otel-collector"],
            "environment": [
                {
                    "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
                    "value": "http://otel-collector:4318/v1/traces"
                }
            ]
        }
    ]
}
```

Migración en Elastic Beanstalk

> **Important**
>
> Detenga el proceso de X-Ray Daemon antes de utilizar el CloudWatch agente para evitar conflictos en los puertos.

La integración actual de X-Ray Daemon se activó al usar la consola de Elastic Beanstalk o mediante la configuración de X-Ray Daemon en el código de origen de la aplicación con un archivo de configuración.

Uso del agente CloudWatch

En la plataforma Amazon Linux 2, configure el CloudWatch agente mediante un archivo `.ebextensions` de configuración:

1. Creación de un directorio llamado `.ebextensions` en la raíz del proyecto
2. Cree un archivo llamado `cloudwatch.config` en el directorio `.ebextensions` con el contenido siguiente:

```
files:
  "/opt/aws/amazon-cloudwatch-agent/etc/config.json":
    mode: "0644"
    owner: root
```

```
    group: root
    content: |
      {
        "traces": {
          "traces_collected": {
            "otlp": {
              "grpc_endpoint": "12.0.0.1:4317",
              "http_endpoint": "12.0.0.1:4318"
            }
          }
        }
      }
container_commands:
  start_agent:
    command: /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a
 append-config -c file:/opt/aws/amazon-cloudwatch-agent/etc/config.json -s
```

3. Incluya el directorio `.ebextensions` en el paquete de código fuente de la aplicación cuando implemente

Para obtener más información sobre los archivos de configuración de Elastic Beanstalk, consulte Personalización avanzada del entorno con archivos de configuración.

Migración a Java OpenTelemetry

Esta sección proporciona orientación sobre la migración del SDK de X-Ray al OpenTelemetry SDK para aplicaciones Java.

Secciones

- Solución de instrumentación automática sin código
- Soluciones de instrumentación manual con el SDK
- Rastreo de las solicitudes entrantes (instrumentación del marco de Spring)
- AWS Instrumentación del SDK v2
- Instrumentación de llamadas a HTTP salientes
- Compatibilidad de instrumentación para otras bibliotecas
- Creación manual de datos de rastros
- Instrumentación de Lambda

Solución de instrumentación automática sin código

With X-Ray Java agent

Para habilitar el agente de X-Ray Java, era necesario modificar los argumentos de JVM de la aplicación.

```
-javaagent:/path-to-disco/disco-java-agent.jar=pluginPath=/path-to-disco/disco-plugins
```

With OpenTelemetry-based Java agent

Para usar agentes de Java OpenTelemetry basados en Java.

- Utilice el agente Java AWS Distro for OpenTelemetry (ADOT) Auto-Instrumentation para la instrumentación automática con el agente Java ADOT. Para obtener más información, consulte Agente de instrumentación automática para rastros y métricas con Java. Si solo desea realizar un seguimiento, desactive la variable de entorno `OTEL_METRICS_EXPORTER=none` para exportar las métricas del agente de Java.

  (Opcional) También puede habilitar CloudWatch Application Signals al instrumentar automáticamente sus aplicaciones AWS con la instrumentación automática de ADOT Java para monitorear el estado actual de las aplicaciones y realizar un seguimiento del rendimiento de las aplicaciones a largo plazo. Las señales de aplicaciones le proporcionan una visión unificada y centrada en las aplicaciones de las aplicaciones, servicios y dependencias y lo ayuda a supervisar y evaluar el estado de las aplicaciones. Para obtener más información, consulte Application Signals.
- Utilice el agente OpenTelemetry Java para la instrumentación automática. Para obtener más información, consulte Instrumentación sin código con el agente de Java.

Soluciones de instrumentación manual con el SDK

Tracing setup with X-Ray SDK

Para instrumentar el código con el X-Ray SDK para Java, primero era necesario configurar la clase de `AWSXRay` con complementos de servicio y reglas de muestreo locales y, a continuación, se utilizó una grabadora proporcionada.

```
static { AWS XRayRecorderBuilder builder = AWS
 XRayRecorderBuilder.standard().withPlugin(new EC2Plugin()).withPlugin(new
 ECSPlugin()); AWS XRay.setGlobalRecorder(builder.build());
}
```

Tracing setup with OpenTelemetry SDK

Se requieren las dependencias siguientes.

```
<dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>io.opentelemetry</groupId>
                <artifactId>opentelemetry-bom</artifactId>
                <version>1.49.0</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>io.opentelemetry.instrumentation</groupId>
                <artifactId>opentelemetry-instrumentation-bom</artifactId>
                <version>2.15.0</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>io.opentelemetry</groupId>
            <artifactId>opentelemetry-sdk</artifactId>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry</groupId>
            <artifactId>opentelemetry-api</artifactId>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.semconv</groupId>
            <artifactId>opentelemetry-semconv</artifactId>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry</groupId>
            <artifactId>opentelemetry-exporter-otlp</artifactId>
```

```
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.contrib</groupId>
            <artifactId>opentelemetry-aws-xray</artifactId>
            <version>1.46.0</version>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.contrib</groupId>
            <artifactId>opentelemetry-aws-xray-propagator</artifactId>
            <version>1.46.0-alpha</version>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.contrib</groupId>
            <artifactId>opentelemetry-aws-resources</artifactId>
            <version>1.46.0-alpha</version>
        </dependency>
    </dependencies>
```

Configure el OpenTelemetry SDK creando una instancia de un objeto `TracerProvider` y registrándolo globalmente. `OpenTelemetrySdk` Configure estos componentes:

- Un exportador OTLP Span (por ejemplo, OtlpGrpcSpanExporter): necesario para exportar las trazas al agente o al recopilador CloudWatch OpenTelemetry
- Un propagador de AWS rayos X: necesario para propagar el contexto de rastreo a AWS los servicios integrados con X-Ray
- Un muestreador remoto de AWS rayos X: obligatorio si necesita muestrear solicitudes utilizando las reglas de muestreo de rayos X
- Detectores de recursos (por ejemplo, EcsResource o Ec2Resource): detectan los metadatos del host que ejecuta la aplicación

```
import io.opentelemetry.api.common.Attributes;
import io.opentelemetry.context.propagation.ContextPropagators;
import io.opentelemetry.contrib.aws.resource.Ec2Resource;
import io.opentelemetry.contrib.aws.resource.EcsResource;
import io.opentelemetry.contrib.awsxray.AwsXrayRemoteSampler;
import io.opentelemetry.contrib.awsxray.propagator.AwsXrayPropagator;
import io.opentelemetry.exporter.otlp.trace.OtlpGrpcSpanExporter;
import io.opentelemetry.sdk.OpenTelemetrySdk;
import io.opentelemetry.sdk.resources.Resource;
import io.opentelemetry.sdk.trace.SdkTracerProvider;
import io.opentelemetry.sdk.trace.export.BatchSpanProcessor;
```

```
import io.opentelemetry.sdk.trace.samplers.Sampler;
import static io.opentelemetry.semconv.ServiceAttributes.SERVICE_NAME;

// ...

    private static final Resource otelResource =
        Resource.create(Attributes.of(SERVICE_NAME, "YOUR_SERVICE_NAME"))
            .merge(EcsResource.get())
            .merge(Ec2Resource.get());
    private static final SdkTracerProvider sdkTracerProvider =
        SdkTracerProvider.builder()
            .addSpanProcessor(BatchSpanProcessor.create(
                OtlpGrpcSpanExporter.getDefault()
            ))
            .addResource(otelResource)
            .setSampler(Sampler.parentBased(
                AwsXrayRemoteSampler.newBuilder(otelResource).build()
            ))
            .build();
    // Globally registering a TracerProvider makes it available throughout the
 application to create as many Tracers as needed.
    private static final OpenTelemetrySdk openTelemetry =
        OpenTelemetrySdk.builder()
            .setTracerProvider(sdkTracerProvider)

 .setPropagators(ContextPropagators.create(AwsXrayPropagator.getInstance()))
            .buildAndRegisterGlobal();
```

Rastreo de las solicitudes entrantes (instrumentación del marco de Spring)

With X-Ray SDK

Para obtener información sobre cómo utilizar el X-Ray SDK con el marco de Spring para instrumentar la aplicación, consulte AOP con Spring y el X-Ray SDK para Java. Para activar AOP en Spring, complete estos pasos.

1. Configure Spring
2. Agregación de un filtro de rastreo a la aplicación
3. Anotación del código o implementación de una interfaz
4. Active X-Ray en la aplicación

With OpenTelemetry SDK

OpenTelemetry proporciona bibliotecas de instrumentación para recopilar el seguimiento de las solicitudes entrantes de las aplicaciones Spring Boot. Para habilitar la instrumentación de Spring Boot con una configuración mínima, incluya la siguiente dependencia.

```
<dependency>
           <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-spring-boot-starter</artifactId>
       </dependency>
```

Para obtener más información sobre cómo habilitar y configurar la instrumentación de Spring Boot para su OpenTelemetry configuración, consulte Primeros OpenTelemetry pasos.

Using OpenTelemetry-based Java agents

El método recomendado por defecto para instrumentar las aplicaciones de Spring Boot es utilizar el agente OpenTelemetry Java con instrumentación de bytecode, que también proporciona más out-of-the-box instrumentaciones y configuraciones en comparación con el uso directo del SDK. Para empezar, consulte Solución de instrumentación automática sin código.

AWS Instrumentación del SDK v2

With X-Ray SDK

El X-Ray SDK for Java puede instrumentar automáticamente todos los clientes del AWS SDK v2 al añadir el `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` submódulo a la compilación.

Para instrumentar las llamadas de clientes individuales posteriores a AWS los servicios con el AWS SDK para Java 2.2 y versiones posteriores, se excluyó el `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` módulo de la configuración de compilación y se incluyó el `aws-xray-recorder-sdk-aws-sdk-v2` módulo. Los clientes individuales se instrumentaron configurándolos con un `TracingInterceptor`.

```
import com.amazonaws.xray.interceptors.TracingInterceptor;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
//...
```

```
public class MyModel {
  private DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_WEST_2)
    .overrideConfiguration(
      ClientOverrideConfiguration.builder()
        .addExecutionInterceptor(new TracingInterceptor())
        .build()
      )
    .build();
//...
```

With OpenTelemetry SDK

Para instrumentar automáticamente todos los clientes AWS del SDK, añade el `opentelemetry-aws-sdk-2.2-autoconfigure` submódulo.

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-aws-sdk-2.2-autoconfigure</artifactId>
            <version>2.15.0-alpha</version>
            <scope>runtime</scope>
        </dependency>
```

Para instrumentar clientes de AWS SDK individuales, añada el `opentelemetry-aws-sdk-2.2` submódulo.

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-aws-sdk-2.2</artifactId>
            <version>2.15.0-alpha</version>
            <scope>compile</scope>
        </dependency>
```

A continuación, registre un interceptor al crear un cliente de AWS SDK.

```
import io.opentelemetry.instrumentation.awssdk.v2_2.AwsSdkTelemetry;

// ...

    AwsSdkTelemetry telemetry = AwsSdkTelemetry.create(openTelemetry);
    private final S3Client S3_CLIENT = S3Client.builder()
```

```
      .overrideConfiguration(ClientOverrideConfiguration.builder()
        .addExecutionInterceptor(telemetry.newExecutionInterceptor())
        .build())
      .build();
```

Instrumentación de llamadas a HTTP salientes

With X-Ray SDK

Para instrumentar las solicitudes HTTP salientes con X-Ray, se necesitaba el SDK de X-Ray para la versión de Apache HttpClient de Java.

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
  public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
```

With OpenTelemetry SDK

Al igual que el SDK de Java X-Ray, OpenTelemetry proporciona una `ApacheHttpClientTelemetry` clase que tiene un método de creación que permite la creación de una instancia de un `HttpClientBuilder` para proporcionar OpenTelemetry intervalos basados y propagación de contexto para Apache HttpClient.

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-apache-httpclient-5.2</artifactId>
            <version>2.15.0-alpha</version>
            <scope>compile</scope>
        </dependency>
```

El siguiente es un ejemplo de código de opentelemetry-java-instrumentation. El cliente HTTP proporcionado por newHttpClient () generará seguimientos para las solicitudes ejecutadas.

```
import io.opentelemetry.api.OpenTelemetry;
import
 io.opentelemetry.instrumentation.apachehttpclient.v5_2.ApacheHttpClientTelemetry;
import org.apache.hc.client5.http.classic.HttpClient;
import org.apache.hc.client5.http.impl.classic.HttpClientBuilder;
```

```
public class ApacheHttpClientConfiguration {

  private OpenTelemetry openTelemetry;

  public ApacheHttpClientConfiguration(OpenTelemetry openTelemetry) {
    this.openTelemetry = openTelemetry;
  }

  // creates a new http client builder for constructing http clients with open
 telemetry instrumentation
  public HttpClientBuilder createBuilder() {
    return
 ApacheHttpClientTelemetry.builder(openTelemetry).build().newHttpClientBuilder();
  }

  // creates a new http client with open telemetry instrumentation
  public HttpClient newHttpClient() {
    return ApacheHttpClientTelemetry.builder(openTelemetry).build().newHttpClient();
  }
}
```

Compatibilidad de instrumentación para otras bibliotecas

Encuentre la lista completa de instrumentaciones de biblioteca compatibles con OpenTelemetry Java en su GitHub repositorio de instrumentación respectivo, en Bibliotecas, marcos, servidores de aplicaciones y. JVMs

También puede buscar en el OpenTelemetry Registro si OpenTelemetry es compatible con la instrumentación. Para empezar a buscar, consulte Registro.

Creación manual de datos de rastros

With X-Ray SDK

Con el X-Ray SDK, se necesitan los métodos `beginSegment` y `beginSubsegment` para crear manualmente segmentos y subsegmentos de X-Ray.

```
  Segment segment = xrayRecorder.beginSegment("ManualSegment");
        segment.putAnnotation("annotationKey", "annotationValue");
        segment.putMetadata("metadataKey", "metadataValue");
```

```
        try {
            Subsegment subsegment =
 xrayRecorder.beginSubsegment("ManualSubsegment");
            subsegment.putAnnotation("key", "value");

            // Do something here

        } catch (Exception e) {
            subsegment.addException(e);
        } finally {
            xrayRecorder.endSegment();
        }
```

With OpenTelemetry SDK

Puede utilizar intervalos personalizados para supervisar el rendimiento de las actividades internas que no se recopilan en las bibliotecas de instrumentación. Tenga en cuenta que solo los servidores de tipo intervalo se convierten en segmentos de X-Ray, todos los demás intervalos se convierten en subsegmentos de X-Ray.

En primer lugar, tendrá que crear un rastreador para generar intervalos, que puede obtener mediante el método `openTelemetry.getTracer`. Esto proporcionará una instancia de rastreador de `TracerProvider` que se registró globalmente en el ejemplo Soluciones de instrumentación manual con el SDK. Puede crear tantas instancias de rastreador como necesite, pero es habitual tener un solo rastreador para toda la aplicación.

```
Tracer tracer = openTelemetry.getTracer("my-app");
```

Puede utilizar el rastreador para crear intervalos.

```
import io.opentelemetry.api.common.AttributeKey;
import io.opentelemetry.api.trace.Span;
import io.opentelemetry.api.trace.SpanKind;
import io.opentelemetry.api.trace.Tracer;
import io.opentelemetry.context.Scope;

...

// SERVER span will become an X-Ray segment
Span span = tracer.spanBuilder("get-token")
  .setKind(SpanKind.SERVER)
```

```
  .setAttribute("key", "value")
  .startSpan();
try (Scope ignored = span.makeCurrent()) {

  span.setAttribute("metadataKey", "metadataValue");
  span.setAttribute("annotationKey", "annotationValue");

  // The following ensures that "annotationKey: annotationValue" is an annotation in
 X-Ray raw data.
  span.setAttribute(AttributeKey.stringArrayKey("aws.xray.annotations"),
 List.of("annotationKey"));

  // Do something here
}

span.end();
```

Los intervalos tienen un tipo predeterminado de INTERNO.

```
// Default span of type INTERNAL will become an X-Ray subsegment
Span span = tracer.spanBuilder("process-header")
  .startSpan();
try (Scope ignored = span.makeCurrent()) {
  doProcessHeader();
}
```

Añadir anotaciones y metadatos a los seguimientos con el SDK OpenTelemetry

En el ejemplo anterior, el método `setAttribute` se utiliza para agregar atributos a cada intervalo. De forma predeterminada, todos los atributos de intervalo se convertirán en metadatos en los datos sin procesar de X-Ray. Para garantizar que un atributo se convierta en una anotación y no en metadatos, en el ejemplo anterior se agrega la clave de ese atributo a la lista del atributo `aws.xray.annotations`. Para obtener más información, consulte Habilitar las anotaciones de X-Ray personalizadas y anotaciones y metadatos.

Con agentes Java OpenTelemetry basados en

Si está utilizando el agente de Java para instrumentar la aplicación de manera automática, necesitará instrumentar en la aplicación de manera manual. Por ejemplo, al código de instrumentos dentro de la aplicación para secciones que no están cubiertas por ninguna biblioteca de instrumentación automática.

Para realizar la instrumentación manual con el agente, es necesario utilizar el artefacto `opentelemetry-api` . La versión del artefacto no puede ser más reciente que la versión del agente.

```
import io.opentelemetry.api.GlobalOpenTelemetry;
import io.opentelemetry.api.trace.Span;

// ...

        Span parentSpan = Span.current();
        Tracer tracer = GlobalOpenTelemetry.getTracer("my-app");
        Span span = tracer.spanBuilder("my-span-name")
            .setParent(io.opentelemetry.context.Context.current().with(parentSpan))
            .startSpan();
        span.end();
```

Instrumentación de Lambda

With X-Ray SDK

Al usar el X-Ray SDK, una vez que Lambda tenga habilitado el rastreo activo, no se requiere ninguna configuración adicional para usar el X-Ray SDK. Lambda creará un segmento que represente la invocación del controlador de Lambda y puede crear subsegmentos o bibliotecas de instrumentos mediante el X-Ray SDK sin necesidad de realizar ninguna configuración adicional.

With OpenTelemetry-based solutions

Autoinstrumentación de capas Lambda: puede instrumentar automáticamente su Lambda con capas Lambda vendidas AWS mediante las siguientes soluciones:

- CloudWatch Capa Lambda de Application Signals (recomendada)

> **Note**
>
> Esta capa Lambda tiene las señales de CloudWatch aplicación habilitadas de forma predeterminada, lo que permite monitorear el rendimiento y el estado de la aplicación Lambda mediante la recopilación de métricas y trazas. Solo para el rastreo, defina la variable de entorno de Lambda `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`.

  - Habilita la supervisión del rendimiento y el estado de la aplicación de Lambda
  - Recopila las métricas y los rastros de forma predeterminada
- AWS capa Lambda gestionada para ADOT Java. Para obtener más información, consulte AWS Distro for OpenTelemetry Lambda Support For Java.

Para utilizar la instrumentación manual junto con la capa de instrumentación automática, consulte Soluciones de instrumentación manual con el SDK. Para reducir los arranques en frío, considere la posibilidad de utilizar instrumentación OpenTelemetry manual para generar OpenTelemetry trazas para la función Lambda.

OpenTelemetry instrumentación manual para Lambda AWS

Considere el siguiente código de función de Lambda que realiza una llamada a Amazon S3 ListBuckets .

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class ListBucketsLambda implements RequestHandler<String, String> {

    private final S3Client S3_CLIENT = S3Client.builder()
        .build();

    @Override
    public String handleRequest(String input, Context context) {
        try {
            ListBucketsResponse response = makeListBucketsCall();
            context.getLogger().log("response: " + response.toString());
            return "Success";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
```

```
    }

    private ListBucketsResponse makeListBucketsCall() {
        try {
            ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder()
                .build();
            ListBucketsResponse response = S3_CLIENT.listBuckets(listBucketsRequest);
            return response;
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to call S3 listBuckets" +
 e.awsErrorDetails().errorMessage(), e);
        }
    }
}
```

A continuación, se muestran las dependencias.

```
dependencies {
    implementation('com.amazonaws:aws-lambda-java-core:1.2.3')
    implementation('software.amazon.awssdk:s3:2.28.29')
    implementation('org.slf4j:slf4j-nop:2.0.16')
}
```

Para instrumentar manualmente el controlador de Lambda y el cliente de Amazon S3, haga lo siguiente.

1. Sustituya las clases de funciones que implementan `RequestHandler` (o RequestStreamHandler) por las que amplían `TracingRequestHandler` (o TracingRequestStreamHandler).
2. Cree una instancia de un objeto TracerProvider y regístrelo globalmente. OpenTelemetrySdk Se TracerProvider recomienda configurarlo con:
   a. Un procesador de intervalos simple con un exportador de intervalos de X-Ray UDP para enviar rastros al punto de conexión de UDP X-Ray de Lambda
   b. Un muestreador ParentBased siempre activo (predeterminado si no está configurado)
   c. Un recurso con service.name establecido en el nombre de la función de Lambda
   d. Un propagador de X-Ray Lambda
3. Cambie el método `handleRequest` a `doHandleRequest` y pase el objeto `OpenTelemetrySdk` a la clase de base.

4. Instrumente el cliente Amazon S3 con la instrumentación del OpenTemetry AWS SDK registrando el interceptor al crear el cliente.

Necesita las siguientes dependencias OpenTelemetry relacionadas.

```
dependencies {
    ...

    implementation("software.amazon.distro.opentelemetry:aws-distro-opentelemetry-xray-
udp-span-exporter:0.1.0")

    implementation(platform('io.opentelemetry.instrumentation:opentelemetry-
instrumentation-bom:2.14.0'))
    implementation(platform('io.opentelemetry:opentelemetry-bom:1.48.0'))

    implementation('io.opentelemetry:opentelemetry-sdk')
    implementation('io.opentelemetry:opentelemetry-api')
    implementation('io.opentelemetry.contrib:opentelemetry-aws-xray-propagator:1.45.0-
alpha')
    implementation('io.opentelemetry.contrib:opentelemetry-aws-resources:1.45.0-alpha')
    implementation('io.opentelemetry.instrumentation:opentelemetry-aws-lambda-
core-1.0:2.14.0-alpha')
    implementation('io.opentelemetry.instrumentation:opentelemetry-aws-sdk-2.2:2.14.0-
alpha')
}
```

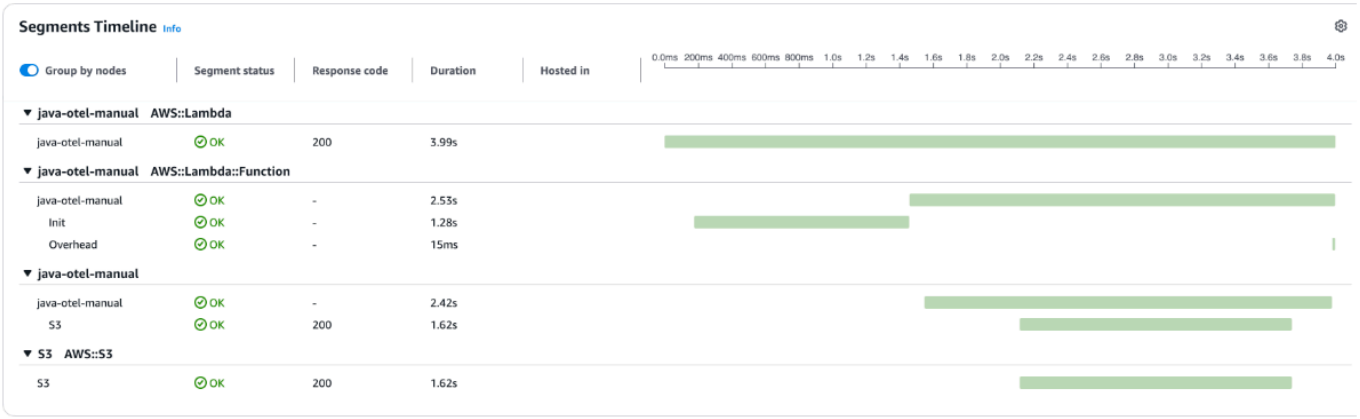
El código siguiente muestra la función de Lambda después de los cambios necesarios. Puede crear intervalos personalizados adicionales para complementar los intervalos que se proporcionan automáticamente.

```
package example;

import java.time.Duration;

import com.amazonaws.services.lambda.runtime.Context;

import io.opentelemetry.api.common.Attributes;
import io.opentelemetry.context.propagation.ContextPropagators;
import io.opentelemetry.contrib.aws.resource.LambdaResource;
import io.opentelemetry.contrib.awsxray.propagator.AwsXrayLambdaPropagator;
import io.opentelemetry.instrumentation.awslambdacore.v1_0.TracingRequestHandler;
import io.opentelemetry.instrumentation.awssdk.v2_2.AwsSdkTelemetry;
```

```
import io.opentelemetry.sdk.OpenTelemetrySdk;
import io.opentelemetry.sdk.resources.Resource;
import io.opentelemetry.sdk.trace.SdkTracerProvider;
import io.opentelemetry.sdk.trace.export.SimpleSpanProcessor;
import io.opentelemetry.sdk.trace.samplers.Sampler;
import static io.opentelemetry.semconv.ServiceAttributes.SERVICE_NAME;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.distro.opentelemetry.exporter.xray.udp.trace.AwsXrayUdpSpanExporterBuilder;

public class ListBucketsLambda extends TracingRequestHandler<String, String> {
    private static final Resource lambdaResource = LambdaResource.get();
    private static final SdkTracerProvider sdkTracerProvider =
        SdkTracerProvider.builder()
            .addSpanProcessor(SimpleSpanProcessor.create(
                new AwsXrayUdpSpanExporterBuilder().build()
            ))
            .addResource(
                lambdaResource
                .merge(Resource.create(Attributes.of(SERVICE_NAME,
 System.getenv("AWS_LAMBDA_FUNCTION_NAME"))))
            )
            .setSampler(Sampler.parentBased(Sampler.alwaysOn()))
            .build();
    private static final OpenTelemetrySdk openTelemetry =
        OpenTelemetrySdk.builder()
            .setTracerProvider(sdkTracerProvider)

 .setPropagators(ContextPropagators.create(AwsXrayLambdaPropagator.getInstance()))
            .buildAndRegisterGlobal();
    private static final AwsSdkTelemetry telemetry =
 AwsSdkTelemetry.create(openTelemetry);
    private final S3Client S3_CLIENT = S3Client.builder()
        .overrideConfiguration(ClientOverrideConfiguration.builder()
            .addExecutionInterceptor(telemetry.newExecutionInterceptor())
            .build())
        .build();

    public ListBucketsLambda() {
        super(openTelemetry, Duration.ofMillis(0));
```

```
    }

    @Override
    public String doHandleRequest(String input, Context context) {
        try {
            ListBucketsResponse response = makeListBucketsCall();
            context.getLogger().log("response: " + response.toString());
            return "Success";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private ListBucketsResponse makeListBucketsCall() {
        try {
            ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder()
                .build();
            ListBucketsResponse response = S3_CLIENT.listBuckets(listBucketsRequest);
            return response;
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to call S3 listBuckets" +
 e.awsErrorDetails().errorMessage(), e);
        }
    }
}
```

Al invocar la función Lambda, verá la siguiente traza en Trace Map en CloudWatch la consola.



Migre a OpenTelemetry Go

Utilice los siguientes ejemplos de código para instrumentar manualmente sus aplicaciones Go con el OpenTelemetry SDK al migrar desde X-Ray.

Instrumentación manual con el SDK

Tracing setup with X-Ray SDK

Al usar el X-Ray SDK para Go, era necesario configurar los complementos de servicio o las reglas de muestreo locales antes de instrumentar el código.

```
func init() {
    if os.Getenv("ENVIRONMENT") == "production" {
        ec2.Init()
    }

    xray.Configure(xray.Config{
        DaemonAddr:       "127.0.0.1:2000",
        ServiceVersion:   "1.2.3",
    })
```

```
}
```

Set up tracing with OpenTelemetry SDK

Configure el OpenTelemetry SDK creando una instancia de un TracerProvider y registrándolo como proveedor de rastreadores globales. Se recomienda configurar los siguientes componentes:

- Exportador de trazas OTLP: necesario para exportar trazas al agente o al recopilador CloudWatch OpenTelemetry
- Propagador de rayos X: necesario para propagar el contexto de la traza a AWS los servicios integrados con X-Ray
- Muestreador remoto de X-Ray: necesario para solicitudes de muestreo con las reglas de muestreo de X-Ray
- Detectores de recursos: para detectar los metadatos del host que ejecuta la aplicación

```
import (
    "go.opentelemetry.io/contrib/detectors/aws/ec2"
    "go.opentelemetry.io/contrib/propagators/aws/xray"
    "go.opentelemetry.io/contrib/samplers/aws/xray"
    "go.opentelemetry.io/otel"
    "go.opentelemetry.io/otel/exporters/otlp/otlptrace/otlptracegrpc"
    "go.opentelemetry.io/otel/sdk/trace"
)

func setupTracing() error {
    ctx := context.Background()

    exporterEndpoint := os.Getenv("OTEL_EXPORTER_OTLP_ENDPOINT")
    if exporterEndpoint == "" {
        exporterEndpoint = "localhost:4317"
    }

    traceExporter, err := otlptracegrpc.New(ctx,
        otlptracegrpc.WithInsecure(),
        otlptracegrpc.WithEndpoint(exporterEndpoint))
    if err != nil {
        return fmt.Errorf("failed to create OTLP trace exporter: %v", err)
    }
```

```
    remoteSampler, err := xray.NewRemoteSampler(ctx, "my-service-name", "ec2")
    if err != nil {
        return fmt.Errorf("failed to create X-Ray Remote Sampler: %v", err)
    }

    ec2Resource, err := ec2.NewResourceDetector().Detect(ctx)
    if err != nil {
        return fmt.Errorf("failed to detect EC2 resource: %v", err)
    }

    tp := trace.NewTracerProvider(
        trace.WithSampler(remoteSampler),
        trace.WithBatcher(traceExporter),
        trace.WithResource(ec2Resource),
    )

    otel.SetTracerProvider(tp)
    otel.SetTextMapPropagator(xray.Propagator{})

    return nil
}
```

Seguimiento de solicitudes entrantes (instrumentación de controlador HTTP)

With X-Ray SDK

Para instrumentar un controlador HTTP con X-Ray, se utilizó el método del controlador de rayos X para generar segmentos utilizando. NewFixedSegmentNamer

```
func main() {
    http.Handle("/", xray.Handler(xray.NewFixedSegmentNamer("myApp"),
 http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Hello!"))
    })))
    http.ListenAndServe(":8000", nil)
}
```

With OpenTelemetry SDK

Para instrumentar un controlador HTTP con OpenTelemetry, usa el método newHandler para empaquetar el código OpenTelemetry del controlador original.

```
import (
    "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"
)

helloHandler := func(w http.ResponseWriter, req *http.Request) {
    ctx := req.Context()
    span := trace.SpanFromContext(ctx)
    span.SetAttributes(attribute.Bool("isHelloHandlerSpan", true),
 attribute.String("attrKey", "attrValue"))

    _, _ = io.WriteString(w, "Hello World!\n")
}

otelHandler := otelhttp.NewHandler(http.HandlerFunc(helloHandler), "Hello")

http.Handle("/hello", otelHandler)
err = http.ListenAndServe(":8080", nil)
if err != nil {
    log.Fatal(err)
}
```

AWS Instrumentación SDK for Go v2

With X-Ray SDK

Para instrumentar AWS las solicitudes salientes del AWS SDK, sus clientes se instrumentaron de la siguiente manera:

```
// Create a segment
ctx, root := xray.BeginSegment(context.TODO(), "AWSSDKV2_Dynamodb")
defer root.Close(nil)

cfg, err := config.LoadDefaultConfig(ctx, config.WithRegion("us-west-2"))
if err != nil {
```

```
    log.Fatalf("unable to load SDK config, %v", err)
}
// Instrumenting AWS SDK v2
awsv2.AWSV2Instrumentor(&cfg.APIOptions)
// Using the Config value, create the DynamoDB client
svc := dynamodb.NewFromConfig(cfg)
// Build the request with its input parameters
_, err = svc.ListTables(ctx, &dynamodb.ListTablesInput{
    Limit: aws.Int32(5),
})
if err != nil {
    log.Fatalf("failed to list tables, %v", err)
}
```

With OpenTelemetry SDK

El AWS SDK for Go v2 Instrumentation proporciona soporte para el rastreo OpenTelemetry de las llamadas descendentes del AWS SDK. A continuación, se muestra un ejemplo de seguimiento de la llamada de un cliente de S3:

```
import (
    ...

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"

    "go.opentelemetry.io/otel"
    oteltrace "go.opentelemetry.io/otel/trace"
    awsConfig "github.com/aws/aws-sdk-go-v2/config"
    "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-sdk-go-v2/
otelaws"
)

...


    // init aws config
    cfg, err := awsConfig.LoadDefaultConfig(ctx)
    if err != nil {
        panic("configuration error, " + err.Error())
    }
```

```
    // instrument all aws clients
    otelaws.AppendMiddlewares(&.APIOptions)


    // Call to S3
    s3Client := s3.NewFromConfig(cfg)
    input := &s3.ListBucketsInput{}
    result, err := s3Client.ListBuckets(ctx, input)
    if err != nil {
        fmt.Printf("Got an error retrieving buckets, %v", err)
        return
    }
```

Instrumentación de llamadas a HTTP salientes

With X-Ray SDK

Para instrumentar las llamadas a HTTP salientes con X-Ray, se utilizó el Xray.Client para crear una copia de un cliente HTTP proporcionado.

```
myClient := xray.Client(http-client)

resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

With OpenTelemetry SDK

Para instrumentar a los clientes HTTP OpenTelemetry, utilice OpenTelemetry otelhttp. NewTransport método para envolver el http. DefaultTransport.

```
import (
    "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"
)

// Create an instrumented HTTP client.
httpClient := &http.Client{
    Transport: otelhttp.NewTransport(
        http.DefaultTransport,
    ),
}
```

```
req, err := http.NewRequestWithContext(ctx, http.MethodGet, "https://api.github.com/
repos/aws-observability/aws-otel-go/releases/latest", nil)
if err != nil {
    fmt.Printf("failed to create http request, %v\n", err)
}
res, err := httpClient.Do(req)
if err != nil {
    fmt.Printf("failed to make http request, %v\n", err)
}
// Request body must be closed
defer func(Body io.ReadCloser) {
    err := Body.Close()
    if err != nil {
        fmt.Printf("failed to close http response body, %v\n", err)
    }
}(res.Body)
```

Compatibilidad de instrumentación para otras bibliotecas

Puede encontrar la lista completa de instrumentaciones de biblioteca compatibles con los paquetes de OpenTelemetry Go under Instrumentation.

También puedes buscar en el OpenTelemetry registro si es OpenTelemetry compatible con la instrumentación de tu biblioteca en el apartado Registro.

Creación manual de datos de rastros

With X-Ray SDK

Con el SDK de X-Ray, se necesitaban BeginSubsegment los métodos BeginSegment y métodos para crear manualmente segmentos y subsegmentos de rayos X.

```
// Start a segment
ctx, seg := xray.BeginSegment(context.Background(), "service-name")
// Start a subsegment
subCtx, subSeg := xray.BeginSubsegment(ctx, "subsegment-name")

// Add metadata or annotation here if necessary
xray.AddAnnotation(subCtx, "annotationKey", "annotationValue")
xray.AddMetadata(subCtx, "metadataKey", "metadataValue")
```

```
subSeg.Close(nil)
// Close the segment
seg.Close(nil)
```

With OpenTelemetry SDK

Use intervalos personalizados para supervisar el rendimiento de las actividades internas que no se recopilan en las bibliotecas de instrumentación. Tenga en cuenta que solo los intervalos de tipo servidor se convierten en segmentos de X-Ray, todos los demás intervalos se convierten en subsegmentos de X-Ray.

En primer lugar, tendrá que crear un rastreador para generar intervalos, que puede obtener mediante el método `otel.Tracer`. Esto proporcionará una instancia de Tracer de la TracerProvider que se registró globalmente en el ejemplo de configuración de rastreo. Puede crear tantas instancias de rastreador como necesite, pero es habitual tener un solo rastreador para toda la aplicación.

```
    tracer := otel.Tracer("application-tracer")
```

```
import (
    ...

    oteltrace "go.opentelemetry.io/otel/trace"
)

...

    var attributes = []attribute.KeyValue{
        attribute.KeyValue{Key: "metadataKey", Value:
 attribute.StringValue("metadataValue")},
        attribute.KeyValue{Key: "annotationKey", Value:
 attribute.StringValue("annotationValue")},
        attribute.KeyValue{Key: "aws.xray.annotations", Value:
 attribute.StringSliceValue([]string{"annotationKey"})},
    }

    ctx := context.Background()

    parentSpanContext, parentSpan := tracer.Start(ctx,
 "ParentSpan", oteltrace.WithSpanKind(oteltrace.SpanKindServer),
 oteltrace.WithAttributes(attributes...))
```

```
    _, childSpan := tracer.Start(parentSpanContext, "ChildSpan",
 oteltrace.WithSpanKind(oteltrace.SpanKindInternal))

    // ...

    childSpan.End()
    parentSpan.End()
```

Añadir anotaciones y metadatos a los seguimientos con el SDK OpenTelemetry

En el ejemplo anterior, el método `WithAttributes` se utiliza para agregar atributos a cada intervalo. Tenga en cuenta que de forma predeterminada, todos los atributos de intervalo se convierten en metadatos en los datos sin procesar de X-Ray. Para garantizar que un atributo se convierta en una anotación y no en metadatos, agregue la clave del atributo a la lista del atributo `aws.xray.annotations`. Para obtener más información, consulte Habilitar anotaciones de X-Ray personalizadas.

Instrumentación manual de Lambda

With X-Ray SDK

Con el X-Ray SDK, una vez que Lambda haya habilitado el rastreo activo, no se requiere ninguna configuración adicional para usar el X-Ray SDK. Lambda ha creado un segmento que representa la invocación del controlador de Lambda y subsegmentos mediante el X-Ray SDK sin necesidad de realizar ninguna configuración adicional.

With OpenTelemetry SDK

El siguiente código de función Lambda (sin instrumentación) realiza una ListBuckets llamada a Amazon S3 y una solicitud HTTP saliente.

```
package main

import (
    "context"
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "os"
```

```
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    awsconfig "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func lambdaHandler(ctx context.Context) (interface{}, error) {
    // Initialize AWS config.
    cfg, err := awsconfig.LoadDefaultConfig(ctx)
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    s3Client := s3.NewFromConfig(cfg)

    // Create an HTTP client.
    httpClient := &http.Client{
        Transport: http.DefaultTransport,
    }

    input := &s3.ListBucketsInput{}
    result, err := s3Client.ListBuckets(ctx, input)
    if err != nil {
        fmt.Printf("Got an error retrieving buckets, %v", err)
    }

    fmt.Println("Buckets:")
    for _, bucket := range result.Buckets {
        fmt.Println(*bucket.Name + ": " + bucket.CreationDate.Format("2006-01-02
 15:04:05 Monday"))
    }
    fmt.Println("End Buckets.")

    req, err := http.NewRequestWithContext(ctx, http.MethodGet, "https://
api.github.com/repos/aws-observability/aws-otel-go/releases/latest", nil)
    if err != nil {
        fmt.Printf("failed to create http request, %v\n", err)
    }
    res, err := httpClient.Do(req)
    if err != nil {
        fmt.Printf("failed to make http request, %v\n", err)
    }
    defer func(Body io.ReadCloser) {
```

```
        err := Body.Close()
        if err != nil {
            fmt.Printf("failed to close http response body, %v\n", err)
        }
    }(res.Body)

    var data map[string]interface{}
    err = json.NewDecoder(res.Body).Decode(&data)
    if err != nil {
        fmt.Printf("failed to read http response body, %v\n", err)
    }
    fmt.Printf("Latest ADOT Go Release is '%s'\n", data["name"])

    return events.APIGatewayProxyResponse{
        StatusCode: http.StatusOK,
        Body:       os.Getenv("_X_AMZN_TRACE_ID"),
    }, nil
}

func main() {
    lambda.Start(lambdaHandler)
}
```

Para instrumentar manualmente el controlador de Lambda y el cliente de Amazon S3, haga lo siguiente:

1. En main (), cree una instancia de a TracerProvider (tp) y regístrelo como proveedor de rastreo global. Se TracerProvider recomienda configurarlo con:
   a. Un procesador de intervalos simple con un exportador de intervalos de X-Ray UDP para enviar rastros al punto de conexión de UDP X-Ray de Lambda
   b. Un recurso con service.name establecido en el nombre de la función de Lambda
2. Cambie el uso de `lambda.Start(lambdaHandler)` a `lambda.Start(otellambda.InstrumentHandler(lambdaHandler, xrayconfig.WithRecommendedOptions(tp)...))`.
3. Instrumente el cliente Amazon S3 con la instrumentación del OpenTemetry AWS SDK añadiendo OpenTelemetry middleware for `aws-sdk-go-v2` a la configuración del cliente de Amazon S3.
4. Instrumente el cliente http utilizando el `otelhttp.NewTransport` método para OpenTelemetry empaquetar el. `http.DefaultTransport`

El código siguiente es un ejemplo del aspecto que tendrá la función de Lambda después de los cambios. Puede crear manualmente intervalos personalizados adicionales además de los intervalos que se proporcionan automáticamente.

```
package main

import (
    "context"
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "os"

    "github.com/aws-observability/aws-otel-go/exporters/xrayudp"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    awsconfig "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"

    lambdadetector "go.opentelemetry.io/contrib/detectors/aws/lambda"
    "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-lambda-go/
otellambda"
    "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-lambda-go/
otellambda/xrayconfig"
    "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-sdk-go-v2/
otelaws"
    "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"
    "go.opentelemetry.io/contrib/propagators/aws/xray"
    "go.opentelemetry.io/otel"
    "go.opentelemetry.io/otel/attribute"
    "go.opentelemetry.io/otel/sdk/resource"
    "go.opentelemetry.io/otel/sdk/trace"
    semconv "go.opentelemetry.io/otel/semconv/v1.26.0"
)

func lambdaHandler(ctx context.Context) (interface{}, error) {
    // Initialize AWS config.
    cfg, err := awsconfig.LoadDefaultConfig(ctx)
    if err != nil {
        panic("configuration error, " + err.Error())
    }
```

```
    // Instrument all AWS clients.
    otelaws.AppendMiddlewares(&cfg.APIOptions)
    // Create an instrumented S3 client from the config.
    s3Client := s3.NewFromConfig(cfg)

    // Create an instrumented HTTP client.
    httpClient := &http.Client{
        Transport: otelhttp.NewTransport(
            http.DefaultTransport,
        ),
    }

    // return func(ctx context.Context) (interface{}, error) {
    input := &s3.ListBucketsInput{}
    result, err := s3Client.ListBuckets(ctx, input)
    if err != nil {
        fmt.Printf("Got an error retrieving buckets, %v", err)
    }

    fmt.Println("Buckets:")
    for _, bucket := range result.Buckets {
        fmt.Println(*bucket.Name + ": " + bucket.CreationDate.Format("2006-01-02
 15:04:05 Monday"))
    }
    fmt.Println("End Buckets.")

    req, err := http.NewRequestWithContext(ctx, http.MethodGet, "https://
api.github.com/repos/aws-observability/aws-otel-go/releases/latest", nil)
    if err != nil {
        fmt.Printf("failed to create http request, %v\n", err)
    }
    res, err := httpClient.Do(req)
    if err != nil {
        fmt.Printf("failed to make http request, %v\n", err)
    }
    defer func(Body io.ReadCloser) {
        err := Body.Close()
        if err != nil {
            fmt.Printf("failed to close http response body, %v\n", err)
        }
    }(res.Body)

    var data map[string]interface{}
    err = json.NewDecoder(res.Body).Decode(&data)
```

```
    if err != nil {
        fmt.Printf("failed to read http response body, %v\n", err)
    }
    fmt.Printf("Latest ADOT Go Release is '%s'\n", data["name"])

    return events.APIGatewayProxyResponse{
        StatusCode: http.StatusOK,
        Body:       os.Getenv("_X_AMZN_TRACE_ID"),
    }, nil
}

func main() {
    ctx := context.Background()
    detector := lambdadetector.NewResourceDetector()
    lambdaResource, err := detector.Detect(context.Background())
    if err != nil {
        fmt.Printf("failed to detect lambda resources: %v\n", err)
    }

    var attributes = []attribute.KeyValue{
        attribute.KeyValue{Key: semconv.ServiceNameKey, Value:
 attribute.StringValue(os.Getenv("AWS_LAMBDA_FUNCTION_NAME"))},
    }
    customResource := resource.NewWithAttributes(semconv.SchemaURL, attributes...)
    mergedResource, _ := resource.Merge(lambdaResource, customResource)

    xrayUdpExporter, _ := xrayudp.NewSpanExporter(ctx)
    tp := trace.NewTracerProvider(
        trace.WithSpanProcessor(trace.NewSimpleSpanProcessor(xrayUdpExporter)),
        trace.WithResource(mergedResource),
    )

    defer func(ctx context.Context) {
        err := tp.Shutdown(ctx)
        if err != nil {
            fmt.Printf("error shutting down tracer provider: %v", err)
        }
    }(ctx)

    otel.SetTracerProvider(tp)
    otel.SetTextMapPropagator(xray.Propagator{})

    lambda.Start(otellambda.InstrumentHandler(lambdaHandler,
 xrayconfig.WithRecommendedOptions(tp)...))
```

```
}
```

Al invocar Lambda, verá la siguiente traza en `Trace Map` CloudWatch la consola:

Migre a Node.js OpenTelemetry

En esta sección se explica cómo migrar sus aplicaciones de Node.js del SDK de X-Ray a OpenTelemetry. Abarca los enfoques de instrumentación automática y manual y proporciona ejemplos específicos de casos de uso comunes.

El X-Ray Node.js SDK le ayuda a instrumentar manualmente las aplicaciones de Node.js para el rastreo. En esta sección se proporcionan ejemplos de código para migrar de X-Ray a la OpenTelemetry instrumentación.

Secciones

- Soluciones de instrumentación automática sin código
- Soluciones de instrumentación manual
- Seguimiento de solicitudes entrantes
- AWS Instrumentación del SDK V3 JavaScript
- Instrumentación de llamadas a HTTP salientes
- Compatibilidad de instrumentación para otras bibliotecas
- Creación manual de datos de rastros
- Instrumentación de Lambda

Soluciones de instrumentación automática sin código

Para rastrear las solicitudes con el X-Ray SDK para Node.js, debe modificar el código de la aplicación. Con OpenTelemetry, puede utilizar soluciones de autoinstrumentación sin código para rastrear las solicitudes.

Instrumentación automática de código cero con instrumentaciones automáticas basadas. OpenTelemetry

1. Uso de la AWS distribución para la instrumentación automática OpenTelemetry (ADOT) para Node.js: para obtener información sobre la instrumentación automática para la aplicación Node.js, consulte Rastreo y métricas con la distribución para la instrumentación automática. AWS OpenTelemetry JavaScript

   (Opcional) También puede habilitar CloudWatch Application Signals al instrumentar automáticamente sus aplicaciones AWS con la JavaScript instrumentación automática de ADOT para monitorear el estado actual de las aplicaciones y realizar un seguimiento del rendimiento de las aplicaciones a largo plazo en comparación con sus objetivos comerciales. Application Signals le proporciona una visión unificada y centrada en las aplicaciones de sus aplicaciones, servicios y dependencias y lo ayuda a supervisar y evaluar el estado de las aplicaciones. Para obtener más información, consulte Application Signals.
2. Uso de la instrumentación automática de OpenTelemetry JavaScript código cero: para obtener información sobre la instrumentación automática con la, consulte instrumentación de código cero. OpenTelemetry JavaScript JavaScript

Soluciones de instrumentación manual

Tracing setup with X-Ray SDK

Cuando se usó el X-Ray SDK para Node.js, el paquete `aws-xray-sdk` era necesario para configurar el X-Ray SDK con complementos de servicio o reglas de muestreo locales antes de usar el SDK para instrumentar el código.

```
var AWSXRay = require('aws-xray-sdk');

AWSXRay.config([AWSXRay.plugins.EC2Plugin,AWSXRay.plugins.ElasticBeanstalkPlugin]);
AWSXRay.middleware.setSamplingRules(<path to file>);
```

Tracing setup with OpenTelemetry SDK

> **Note**
>
> AWS El muestreo remoto de rayos X actualmente no está disponible para configurarse para OpenTelemetry JS. Sin embargo, la compatibilidad para el muestreo remoto de X-Ray está disponible actualmente a través de la instrumentación automática de ADOT para Node.js.

Para el ejemplo de código siguiente, necesitará las siguientes dependencias:

```
npm install --save \
  @opentelemetry/api \
  @opentelemetry/sdk-node \
  @opentelemetry/exporter-trace-otlp-proto \
  @opentelemetry/propagator-aws-xray \
  @opentelemetry/resource-detector-aws
```

Debe instalar y configurar el OpenTelemetry SDK antes de ejecutar el código de la aplicación. Esto se puede hacer mediante el indicador –-require. Cree un archivo denominado instrumentation.js, que contendrá la configuración y la configuración de su OpenTelemetry instrumentación.

Se recomienda que configure los siguientes componentes:

- OTLPTraceExportador: necesario para exportar las trazas al agente/colector CloudWatch OpenTelemetry
- AWSXRayPropagador: necesario para propagar el contexto de rastreo a los AWS servicios que están integrados con X-Ray
- Detectores de recursos (por ejemplo, Amazon EC2 Resource Detector): para detectar los metadatos del host que ejecuta la aplicación

```
/*instrumentation.js*/
// Require dependencies
const { NodeSDK } = require('@opentelemetry/sdk-node');
const { OTLPTraceExporter } = require('@opentelemetry/exporter-trace-otlp-proto');
const { AWSXRayPropagator } = require("@opentelemetry/propagator-aws-xray");
const { detectResources } = require('@opentelemetry/resources');
const { awsEc2Detector } = require('@opentelemetry/resource-detector-aws');

const resource = detectResources({
    detectors: [awsEc2Detector],
});

const _traceExporter = new OTLPTraceExporter({
    url: 'http://localhost:4318/v1/traces'
});

const sdk = new NodeSDK({
    resource: resource,
    textMapPropagator: new AWSXRayPropagator(),
    traceExporter: _traceExporter
});

sdk.start();
```

A continuación, puede ejecutar la aplicación con su OpenTelemetry configuración, de la siguiente manera:

```
node --require ./instrumentation.js app.js
```

Puedes usar las instrumentaciones de las bibliotecas del OpenTelemetry SDK para crear automáticamente intervalos para bibliotecas como el SDK. AWS Al habilitarlas, se crearán automáticamente intervalos para módulos como el SDK de la AWS versión 3. JavaScript OpenTelemetry ofrece la opción de habilitar todas las instrumentaciones de la biblioteca o especificar qué instrumentaciones de la biblioteca se deben habilitar.

Para habilitar todas las instrumentaciones, instale el paquete `@opentelemetry/auto-instrumentations-node`:

```
npm install @opentelemetry/auto-instrumentations-node
```

A continuación, actualice la configuración para habilitar todas las instrumentaciones de la biblioteca, como se muestra a continuación.

```
const { getNodeAutoInstrumentations } = require('@opentelemetry/auto-
instrumentations-node');

...

const sdk = new NodeSDK({
    resource: resource,
     instrumentations: [getNodeAutoInstrumentations()],
     textMapPropagator: new AWSXRayPropagator(),
    traceExporter: _traceExporter
});
```

Tracing setup with ADOT auto-instrumentation for Node.js

Puede utilizar la instrumentación automática de ADOT para Node.js OpenTelemetry para configurar automáticamente sus aplicaciones de Node.js. Al utilizar la instrumentación automática de ADOT, no es necesario realizar cambios manuales en el código para rastrear las solicitudes entrantes ni para rastrear bibliotecas como el SDK o los clientes HTTP. AWS Para obtener más información, consulte Seguimiento y métricas con la distribución para la AWS instrumentación automática. OpenTelemetry JavaScript

La instrumentación automática de ADOT para Node.js admite:

- Muestreo remoto de X-Ray a través de una variable de entorno: `export OTEL_TRACES_SAMPLER=xray`
- Propagación de contexto de rastros de X-Ray (habilitada de forma predeterminada)
- Detección de recursos (la detección de recursos para los entornos de Amazon EC2, Amazon ECS y Amazon EKS está habilitada de forma predeterminada)
- Instrumentaciones de biblioteca automáticas para todas las OpenTelemetry instrumentaciones compatibles, que se pueden utilizar de disabled/enabled forma selectiva y variables de entorno `OTEL_NODE_ENABLED_INSTRUMENTATIONS` `OTEL_NODE_DISABLED_INSTRUMENTATIONS`
- Creación manual de intervalos

Seguimiento de solicitudes entrantes

With X-Ray SDK

Express.js

Con el X-Ray SDK para rastrear las solicitudes HTTP entrantes recibidas por las aplicaciones de Express.js, los dos middlewares `AWSXRay.express.openSegment(<name>)` y `AWSXRay.express.closeSegment()` eran necesarios para envolver todas las rutas definidas para rastrearlas.

```
app.use(xrayExpress.openSegment('defaultName'));

...

app.use(xrayExpress.closeSegment());
```

Restify

Para rastrear las solicitudes HTTP entrantes recibidas por las aplicaciones de `Restify`, se utilizó el middleware del X-Ray SDK ejecutando habilitar desde el módulo de `aws-xray-sdk-restify` del servidor de Restify:

```
var AWSXRay = require('aws-xray-sdk');
var AWSXRayRestify = require('aws-xray-sdk-restify');

var restify = require('restify');
var server = restify.createServer();
AWSXRayRestify.enable(server, 'MyApp'));
```

With OpenTelemetry SDK

Express.js

La instrumentación HTTP y la instrumentación express proporcionan soporte para el `Express.js` rastreo de las OpenTelemetry solicitudes entrantes. OpenTelemetry Instale las siguientes dependencias con `npm`:

```
npm install --save @opentelemetry/instrumentation-http @opentelemetry/
instrumentation-express
```

Actualice la configuración del OpenTelemetry SDK para habilitar la instrumentación del módulo express:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
const { ExpressInstrumentation } = require('@opentelemetry/instrumentation-
express');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    // Express instrumentation requires HTTP instrumentation
    new HttpInstrumentation(),
    new ExpressInstrumentation(),
  ],
});
```

Restify

Para las aplicaciones de Restify, necesitará la instrumentación de OpenTelemetry Restify. Instale la siguiente dependencia:

```
npm install --save @opentelemetry/instrumentation-restify
```

Actualice la configuración del OpenTelemetry SDK para habilitar la instrumentación del módulo Restify:

```
const { RestifyInstrumentation } = require('@opentelemetry/instrumentation-
restify');
```

```
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new RestifyInstrumentation(),
  ],
});
```

AWS Instrumentación del SDK V3 JavaScript

With X-Ray SDK

Para instrumentar AWS las solicitudes salientes del AWS SDK, instrumentaste los clientes como en el siguiente ejemplo:

```
import { S3, PutObjectCommand } from '@aws-sdk/client-s3';

const s3 = AWSXRay.captureAWSv3Client(new S3({}));

await s3.send(new PutObjectCommand({
  Bucket: bucketName,
  Key: keyName,
  Body: 'Hello!',
}));
```

With OpenTelemetry SDK

La instrumentación del AWS SDK proporciona soporte para rastrear las llamadas descendentes del SDK a DynamoDB, Amazon S3 y otros. OpenTelemetry AWS Instale la siguiente dependencia con `npm`:

```
npm install --save @opentelemetry/instrumentation-aws-sdk
```

Actualice la configuración del OpenTelemetry SDK con la instrumentación del SDK. AWS

```
import { AwsInstrumentation } from '@opentelemetry/instrumentation-aws-sdk';
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new AwsInstrumentation()
  ],
});
```

Instrumentación de llamadas a HTTP salientes

With X-Ray SDK

Para instrumentar las solicitudes HTTP salientes con X-Ray, era necesario instrumentar los clientes. Por ejemplo, consulte lo siguiente.

Clientes HTTP individuales

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Todos los clientes HTTP (global)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

With OpenTelemetry SDK

HTTP Instrumentation proporciona soporte de rastreo para los clientes OpenTelemetry HTTP de Node.js. Instale la siguiente dependencia con `npm`:

```
npm install --save @opentelemetry/instrumentation-http
```

Actualice la configuración del OpenTelemetry SDK de la siguiente manera:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new HttpInstrumentation(),
  ],
});
```

Compatibilidad de instrumentación para otras bibliotecas

Encontrará la lista completa de instrumentaciones de biblioteca compatibles en Instrumentaciones OpenTelemetry JavaScript compatibles.

También puede buscar en el OpenTelemetry registro si es OpenTelemetry compatible con la instrumentación de su biblioteca en el Registro.

Creación manual de datos de rastros

With X-Ray SDK

Con X-Ray, se necesitaba el código del paquete de `aws-xray-sdk` para crear segmentos manualmente y sus subsegmentos secundarios a fin de rastrear la aplicación.

```
var AWSXRay = require('aws-xray-sdk');

AWSXRay.enableManualMode();
```

```
var segment = new AWSXRay.Segment('myApplication');

captureFunc('1', function(subsegment1) {
  captureFunc('2', function(subsegment2) {

  }, subsegment1);
}, segment);

segment.close();
segment.flush();
```

With OpenTelemetry SDK

Puede crear y usar intervalos personalizados para supervisar el rendimiento de las actividades internas que no se recopilan en las bibliotecas de instrumentación. Tenga en cuenta que solo los intervalos de tipo servidor se convierten en segmentos de X-Ray, todos los demás intervalos se convierten en subsegmentos de X-Ray. Para obtener más información, consulte Segmentos.

Necesitará una instancia de Tracer después de configurar el OpenTelemetry SDK en Tracing Setup para crear Spans. Puede crear tantas instancias de rastreador como necesite, pero es habitual tener un solo rastreador para toda la aplicación.

```
const { trace, SpanKind } = require('@opentelemetry/api');

// Get a tracer instance
const tracer = trace.getTracer('your-tracer-name');

...

  // This span will appear as a segment in X-Ray
  tracer.startActiveSpan('server', { kind: SpanKind.SERVER }, span => {
    // Do work here

    // This span will appear as a subsegment in X-Ray
    tracer.startActiveSpan('operation2', { kind: SpanKind.INTERNAL }, innerSpan => {
      // Do more work here

      innerSpan.end();
    });
    span.end();
  });
```

Añadir anotaciones y metadatos a los seguimientos con el SDK OpenTelemetry

También puede agregar pares clave-valor personalizados como atributos en los intervalos. Tenga en cuenta que de forma predeterminada, todos estos atributos de intervalo se convertirán en metadatos en los datos sin procesar de X-Ray. Para garantizar que un atributo se convierta en una anotación y no en metadatos, agregue la clave del atributo a la lista del atributo `aws.xray.annotations`. Para obtener más información, consulte Habilitar anotaciones de X-Ray personalizadas.

```
 tracer.startActiveSpan('server', { kind: SpanKind.SERVER }, span => {
   span.setAttribute('metadataKey', 'metadataValue');
   span.setAttribute('annotationKey', 'annotationValue');

   // The following ensures that "annotationKey: annotationValue" is an annotation
in X-Ray raw data.
   span.setAttribute('aws.xray.annotations', ['annotationKey']);

   // Do work here

   span.end();
 });
```

Instrumentación de Lambda

With X-Ray SDK

Tras habilitar el rastreo activo para la función de Lambda, se necesitaba el X-Ray SDK sin necesidad de configuración adicional. Lambda crea un segmento que representa la invocación del controlador de Lambda y usted creó subsegmentos o bibliotecas de instrumentos mediante el X-Ray SDK sin necesidad de realizar ninguna configuración adicional.

With OpenTelemetry SDK

Puede instrumentar automáticamente su Lambda con capas Lambda AWS vendidas. Hay dos soluciones:

- Capa lambda (recomendada) de CloudWatch Application Signals

> **Note**
>
> Esta capa Lambda tiene las señales de CloudWatch aplicación habilitadas de forma predeterminada, lo que permite monitorear el rendimiento y el estado de la aplicación Lambda mediante la recopilación de métricas y trazas. Si solo desea realizar un seguimiento, debe configurar la variable de entorno de Lambda `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`. Para obtener más información, consulte Habilitar las aplicaciones en Lambda.

- AWS capa Lambda gestionada para ADOT JS. Para obtener más información, consulte AWS Distro for OpenTelemetry Lambda Support JavaScript For.

Creación manual de intervalos con instrumentación de Lambda

Si bien la capa JavaScript Lambda de ADOT proporciona instrumentación automática para la función Lambda, es posible que necesite realizar una instrumentación manual en la Lambda, por ejemplo, para proporcionar datos personalizados o para instrumentar código dentro de la propia función Lambda que no está cubierto por las instrumentaciones de la biblioteca.

Para realizar la instrumentación manual junto con la instrumentación automática, tendrá que agregar `@opentelemetry/api` como una dependencia. Se recomienda que la versión de esta dependencia sea la misma versión de la misma dependencia que utiliza el SDK de ADOT. JavaScript Puede usar la OpenTelemetry API para crear intervalos manualmente en la función Lambda.

Para agregar la dependencia `@opentelemetry/api` mediante NPM:

```
npm install @opentelemetry/api
```

Migre a.NET OpenTelemetry

Al utilizar el rastreo de X-Ray en las aplicaciones .NET, se utiliza X-Ray .NET SDK con esfuerzos manuales para la instrumentación.

Esta sección proporciona ejemplos de código en la Soluciones de instrumentación manual con el SDK sección para migrar de la solución de instrumentación manual de X-Ray a las soluciones de instrumentación OpenTelemetry manual para.NET. Como alternativa, puede migrar de la instrumentación manual de X-Ray a las soluciones de instrumentación OpenTelemetry automática y a las aplicaciones .NET de los instrumentos sin tener que modificar el código fuente de la aplicación en la Soluciones de instrumentación automática sin código sección.

Secciones

Soluciones de instrumentación automática sin código

OpenTelemetry proporciona soluciones de autoinstrumentación sin código. Estas soluciones rastrean las solicitudes sin requerir cambios en el código de la aplicación.

OpenTelemetryopciones de instrumentación automática basadas en

1. Uso de la AWS distribución para la instrumentación automática OpenTelemetry (ADOT) para .NET: para instrumentar automáticamente las aplicaciones.NET, consulte Seguimiento y métricas con la distribución para la AWS instrumentación automática de.NET. OpenTelemetry

   (Opcional) Habilite las señales de CloudWatch aplicación al instrumentar automáticamente sus aplicaciones AWS con la instrumentación automática de ADOT.NET para:

   - Supervisión del estado actual de la aplicación
   - Realización de un seguimiento del rendimiento de las aplicaciones a largo plazo en comparación con los objetivos empresariales
   - Obtención de una visión unificada y centrada en las aplicaciones de las aplicaciones, servicios y dependencias

- Supervisión y clasificación del estado de las aplicaciones

Para obtener más información, consulte Application Signals.

2. Uso de la instrumentación automática de código cero de OpenTelemetry .Net: para instrumentar automáticamente con .Net, consulte Tracing and Metrics with the Distro para ver la instrumentación automática de OpenTelemetry .NET. AWS OpenTelemetry

Soluciones de instrumentación manual con el SDK

Tracing configuration with X-Ray SDK

Para las aplicaciones web .NET, el X-Ray SDK se configura en la sección appSettings del archivo `Web.config`.

Web.config de ejemplo

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
  </appSettings>
</configuration>
```

En el caso de .NET Core, se utiliza un archivo llamado `appsettings.json` con una clave de nivel superior llamada `XRay` y, a continuación, se crea un objeto de configuración o se inicializa la grabadora de X-Ray.

Ejemplo para .NET `appsettings.json`

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin"
  }
}
```

Ejemplo para .NET Core Program.cs: configuración de grabadora

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Tracing configuration with OpenTelemetry SDK

Agregue estas dependencias:

```
dotnet add package OpenTelemetry
dotnet add package OpenTelemetry.Contrib.Extensions.AWSXRay
dotnet add package OpenTelemetry.Sampler.AWS --prerelease
dotnet add package OpenTelemetry.Resources.AWS
dotnet add package OpenTelemetry.Exporter.OpenTelemetryProtocol
dotnet add package OpenTelemetry.Extensions.Hosting
dotnet add package OpenTelemetry.Instrumentation.AspNetCore
```

Para su aplicación.NET, configure el OpenTelemetry SDK configurando el Global TracerProvider. La configuración de ejemplo siguiente también habilita la instrumentación para `ASP.NET Core`. Para instrumentar `ASP.NET`, consulte Rastreo de las solicitudes entrantes (ASP.NET e instrumentación básica de ASP.NET). Para usarlo OpenTelemetry con otros marcos, consulte el Registro para ver más bibliotecas de marcos compatibles.

Se recomienda que configure los siguientes componentes:

- `An OTLP Exporter`— Necesario para exportar trazas al CloudWatch OpenTelemetry agente/recopilador
- Un propagador de AWS rayos X: necesario para propagar el contexto de rastreo a AWS los servicios que están integrados con X-Ray
- Un muestreador remoto de AWS rayos X: necesario si necesita muestrear solicitudes utilizando las reglas de muestreo de rayos X
- `Resource Detectors`(por ejemplo, Amazon EC2 Resource Detector): para detectar los metadatos del host que ejecuta la aplicación

```
using OpenTelemetry;
using OpenTelemetry.Contrib.Extensions.AWSXRay.Trace;
using OpenTelemetry.Sampler.AWS;
using OpenTelemetry.Trace;
using OpenTelemetry.Resources;

var builder = WebApplication.CreateBuilder(args);

var serviceName = "MyServiceName";
var serviceVersion = "1.0.0";

var resourceBuilder = ResourceBuilder
    .CreateDefault()
    .AddService(serviceName: serviceName)
    .AddAWSEC2Detector();

builder.Services.AddOpenTelemetry()
    .ConfigureResource(resource => resource
        .AddAWSEC2Detector()
        .AddService(
            serviceName: serviceName,
            serviceVersion: serviceVersion))
    .WithTracing(tracing => tracing
        .AddSource(serviceName)
        .AddAspNetCoreInstrumentation()
        .AddOtlpExporter()
        .SetSampler(AWSXRayRemoteSampler.Builder(resourceBuilder.Build())
            .SetEndpoint("http://localhost:2000")
            .Build()));

Sdk.SetDefaultTextMapPropagator(new AWSXRayPropagator()); // configure  X-Ray
 propagator
```

OpenTelemetry Para utilizarla en una aplicación de consola, añada la siguiente OpenTelemetry configuración al iniciar el programa.

```
using OpenTelemetry;
using OpenTelemetry.Contrib.Extensions.AWSXRay.Trace;
using OpenTelemetry.Trace;
using OpenTelemetry.Resources;
```

```
var serviceName = "MyServiceName";

var resourceBuilder = ResourceBuilder
    .CreateDefault()
    .AddService(serviceName: serviceName)
    .AddAWSEC2Detector();

var tracerProvider = Sdk.CreateTracerProviderBuilder()
    .AddSource(serviceName)
    .ConfigureResource(resource =>
        resource
            .AddAWSEC2Detector()
            .AddService(
                serviceName: serviceName,
                serviceVersion: serviceVersion
            )
        )
    .AddOtlpExporter() // default address localhost:4317
    .SetSampler(new TraceIdRatioBasedSampler(1.00))
    .Build();

Sdk.SetDefaultTextMapPropagator(new AWSXRayPropagator()); // configure  X-Ray
 propagator
```

Creación manual de datos de rastros

With X-Ray SDK

Con el X-Ray SDK, se necesitaban los métodos `BeginSegment` y `BeginSubsegment` para crear manualmente segmentos y subsegmentos de X-Ray.

```
using Amazon.XRay.Recorder.Core;

AWSXRayRecorder.Instance.BeginSegment("segment name"); // generates `TraceId` for
 you
try
{
    // Do something here
    // can create custom subsegments
    AWSXRayRecorder.Instance.BeginSubsegment("subsegment name");
```

```
    try
    {
        DoSometing();
    }
    catch (Exception e)
    {
        AWSXRayRecorder.Instance.AddException(e);
    }
    finally
    {
        AWSXRayRecorder.Instance.EndSubsegment();
    }
}
catch (Exception e)
{
    AWSXRayRecorder.Instance.AddException(e);
}
finally
{
    AWSXRayRecorder.Instance.EndSegment();
}
```

With OpenTelemetry SDK

En .NET, puede usar la API de actividad para crear intervalos personalizados para supervisar el rendimiento de las actividades internas que no se recopilan en las bibliotecas de instrumentación. Tenga en cuenta que solo los intervalos de tipo servidor se convierten en segmentos de X-Ray, todos los demás intervalos se convierten en subsegmentos de X-Ray.

Puede crear tantas instancias de `ActivitySource` como necesite, pero es habitual tener un solo rastreador para toda la aplicación o servicio.

```
using System.Diagnostics;

ActivitySource activitySource = new ActivitySource("ActivitySourceName",
 "ActivitySourceVersion");


...
```

```
using (var activity = activitySource.StartActivity("ActivityName",
 ActivityKind.Server)) // this will be translated to a X-Ray Segment
{
    // Do something here

    using (var internalActivity = activitySource.StartActivity("ActivityName",
 ActivityKind.Internal)) // this will be translated to an X-Ray Subsegment
    {
        // Do something here
    }
}
```

Añadir anotaciones y metadatos a los seguimientos con OpenTelemetry el SDK

También puede agregar pares clave-valor personalizados como atributos en los intervalos con el método `SetTag` en una actividad. Tenga en cuenta que de forma predeterminada, todos los atributos de intervalo se convertirán en metadatos en los datos sin procesar de X-Ray. Para garantizar que un atributo se convierta en una anotación y no en metadatos, puede agregar esa clave del atributo a la lista del atributo `aws.xray.annotations`.

```
using (var activity = activitySource.StartActivity("ActivityName",
 ActivityKind.Server)) // this will be translated to a X-Ray Segment
{
    activity.SetTag("metadataKey", "metadataValue");
    activity.SetTag("annotationKey", "annotationValue");
    string[] annotationKeys = {"annotationKey"};
    activity.SetTag("aws.xray.annotations", annotationKeys);

    // Do something here

    using (var internalActivity = activitySource.StartActivity("ActivityName",
 ActivityKind.Internal)) // this will be translated to an X-Ray Subsegment
    {
        // Do something here
    }
}
```

Con instrumentación OpenTelemetry automática

Si utiliza una solución de instrumentación OpenTelemetry automática para.NET y necesita realizar instrumentación manual en su aplicación, por ejemplo, instrumentar código dentro de la propia aplicación para secciones que no están incluidas en ninguna biblioteca de autoinstrumentación.

Como solo puede haber un `TracerProvider` global, la instrumentación manual no debe crear una instancia propia de `TracerProvider` si se usa junto con la instrumentación automática. Cuando `TracerProvider` se usa, el rastreo manual personalizado funciona de la misma manera cuando se usa instrumentación automática o instrumentación manual a través del SDK. OpenTelemetry

Rastreo de las solicitudes entrantes (ASP.NET e instrumentación básica de ASP.NET)

With X-Ray SDK

Para instrumentar las solicitudes atendidas por la aplicación de ASP.NET, consulte https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-dotnet-messagehandler.html para obtener información sobre cómo llamar a `RegisterXRay` en el método `Init` del archivo `global.asax`.

```
AWSXRayASPNET.RegisterXRay(this, "MyApp");
```

Para instrumentar las solicitudes atendidas por la aplicación principal de ASP.NET, se llama al método `UseXRay` antes que a cualquier otro middleware del método `Configure` de la clase Startup.

```
app.UseXRay("MyApp");
```

With OpenTelemetry SDK

OpenTelemetry también proporciona bibliotecas de instrumentación para recopilar los seguimientos de las solicitudes web entrantes de ASP.NET y ASP.NET core. En la siguiente sección se enumeran los pasos necesarios para agregar y habilitar estas instrumentaciones de

biblioteca para su OpenTelemetry configuración, incluida la forma de agregar la instrumentación principal de ASP.NET o ASP.NET al crear el proveedor Tracer.

Para obtener información sobre cómo habilitar .Instrumentation. OpenTelemetry AspNet, consulte Pasos para habilitar OpenTelemetry .Instrumentation. AspNety para obtener información sobre cómo habilitar OpenTelemetry .Instrumentation. AspNetCore, consulte los pasos para habilitar OpenTelemetry .Instrumentation. AspNetCore.

AWS Instrumentación del SDK

With X-Ray SDK

Instala todos los clientes del AWS SDK llamando`RegisterXRayForAllServices()`.

```
using Amazon.XRay.Recorder.Handlers.AwsSdk;
AWSSDKHandler.RegisterXRayForAllServices(); //place this before any instantiation of
 AmazonServiceClient
AmazonDynamoDBClient client = new AmazonDynamoDBClient(RegionEndpoint.USWest2); //
 AmazonDynamoDBClient is automatically registered with X-Ray
```

Utilice uno de los siguientes métodos para la instrumentación específica AWS del cliente de servicio.

```
AWSSDKHandler.RegisterXRay<IAmazonDynamoDB>(); // Registers specific type of
 AmazonServiceClient : All instances of IAmazonDynamoDB created after this line are
 registered
AWSSDKHandler.RegisterXRayManifest(String path); // To configure custom AWS Service
 Manifest file. This is optional, if you have followed "Configuration" section
```

With OpenTelemetry SDK

Para el ejemplo de código siguiente, necesitará la siguiente dependencia:

```
dotnet add package OpenTelemetry.Instrumentation.AWS
```

Para instrumentar el AWS SDK, actualice la configuración del OpenTelemetry SDK donde TracerProvider está configurado el Global.

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
        .AddAWSInstrumentation()
        ...
```

Instrumentación de llamadas a HTTP salientes

With X-Ray SDK

El X-Ray .NET SDK rastrea las llamadas HTTP salientes a través de los métodos de extensión `GetResponseTraced()` o `GetAsyncResponseTraced()` cuando se usa `System.Net.HttpWebRequest`, o mediante el controlador `HttpClientXRayTracingHandler` cuando se usa `System.Net.Http.HttpClient`.

With OpenTelemetry SDK

Para el ejemplo de código siguiente, necesitará la siguiente dependencia:

```
dotnet add package OpenTelemetry.Instrumentation.Http
```

Para `System.Net.Http.HttpClient` instrumentar y `System.Net.HttpWebRequest` actualizar la configuración del OpenTelemetry SDK donde TracerProvider está configurado el Global.

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
        .AddHttpClientInstrumentation()
        ...
```

Compatibilidad de instrumentación para otras bibliotecas

Puede buscar y filtrar en el OpenTelemetry Registro las bibliotecas de instrumentación.NET para averiguar si son OpenTelemetry compatibles con la instrumentación de su biblioteca. Consulte Registro para empezar a buscar.

Instrumentación de Lambda

With X-Ray SDK

Para utilizar X-Ray SDK con Lambda se requirió el siguiente procedimiento:

1. Habilitación del seguimiento activo en la función de Lambda
2. El servicio de Lambda crea un segmento que representa la invocación del controlador
3. Creación de subsegmentos o bibliotecas de instrumentos con el X-Ray SDK

With OpenTelemetry-based solutions

Puede instrumentar automáticamente su Lambda con capas Lambda AWS vendidas. Hay dos soluciones:

- Capa lambda (recomendada) de CloudWatch Application Signals
- Para obtener un mejor rendimiento, puede considerar utilizarla `OpenTelemetry Manual Instrumentation` para generar OpenTelemetry trazas para la función Lambda.

OpenTelemetry instrumentación manual para Lambda AWS

A continuación, se muestra el ejemplo del código de la función de Lambda (sin instrumentación).

```
using System;
using System.Text;
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using Amazon.S3.Model;

// Assembly attribute to enable Lambda function logging
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer
```

```
namespace ExampleLambda;

public class ListBucketsHandler
{
    private static readonly AmazonS3Client s3Client = new();

    // new Lambda function handler passed in
    public async Task<string> HandleRequest(object input, ILambdaContext context)
    {
        try
        {
            var DoListBucketsAsyncResponse = await DoListBucketsAsync();
            context.Logger.LogInformation($"Results:
 {DoListBucketsAsyncResponse.Buckets}");

            context.Logger.LogInformation($"Successfully called ListBucketsAsync");
            return "Success!";
        }
        catch (Exception ex)
        {
            context.Logger.LogError($"Failed to call ListBucketsAsync: {ex.Message}");
            throw;
        }
    }

    private async Task<ListBucketsResponse> DoListBucketsAsync()
    {
        try
        {
            var putRequest = new ListBucketsRequest
            {
            };

            var response = await s3Client.ListBucketsAsync(putRequest);
            return response;
        }
        catch (AmazonS3Exception ex)
        {
            throw new Exception($"Failed to call ListBucketsAsync: {ex.Message}", ex);
        }
    }
}
```

Para instrumentar manualmente el controlador de Lambda y el cliente de Amazon S3, haga lo siguiente.

1. Instanciar a TracerProvider : TracerProvider se recomienda configurar con un`XrayUdpSpanExporter`, un muestreador ParentBased Always On y un `Resource` con `service.name` establecido en el nombre de la función Lambda.
2. Instrumente el cliente Amazon S3 con la instrumentación del OpenTemetry AWS SDK llamando `AddAWSInstrumentation()` para añadir la instrumentación AWS del cliente del SDK a `TracerProvider`
3. Cree una función envolvente con la misma firma que la función de Lambda original. Llame a la API de `AWSLambdaWrapper.Trace()` y pase `TracerProvider`, la función de Lambda original y sus entradas como parámetros. Configure la función envolvente como entrada del controlador de Lambda.

Para el ejemplo de código siguiente, necesitará las siguientes dependencias:

```
dotnet add package OpenTelemetry.Instrumentation.AWSLambda
dotnet add package OpenTelemetry.Instrumentation.AWS
dotnet add package OpenTelemetry.Resources.AWS
dotnet add package AWS.Distro.OpenTelemetry.Exporter.Xray.Udp
```

El código siguiente muestra la función de Lambda después de los cambios necesarios. Puede crear intervalos personalizados adicionales para complementar los intervalos que se proporcionan automáticamente.

```
using Amazon.Lambda.Core;
using Amazon.S3;
using Amazon.S3.Model;
using OpenTelemetry;
using OpenTelemetry.Instrumentation.AWSLambda;
using OpenTelemetry.Trace;
using AWS.Distro.OpenTelemetry.Exporter.Xray.Udp;
using OpenTelemetry.Resources;
```

```
// Assembly attribute to enable Lambda function logging
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer

namespace ExampleLambda;

public class ListBucketsHandler
{
    private static readonly AmazonS3Client s3Client = new();

    TracerProvider tracerProvider = Sdk.CreateTracerProviderBuilder()
        .AddAWSLambdaConfigurations()
        .AddProcessor(
            new SimpleActivityExportProcessor(
                // AWS_LAMBDA_FUNCTION_NAME Environment Variable will be defined in AWS
 Lambda Environment
                new
 XrayUdpExporter(ResourceBuilder.CreateDefault().AddService(Environment.GetEnvironmentVariable(
            )
        )
        .AddAWSInstrumentation()
        .SetSampler(new ParentBasedSampler(new AlwaysOnSampler()))
        .Build();

    // new Lambda function handler passed in
    public async Task<string> HandleRequest(object input, ILambdaContext context)
    => await AWSLambdaWrapper.Trace(tracerProvider, OriginalHandleRequest, input,
 context);

    public async Task<string> OriginalHandleRequest(object input, ILambdaContext
 context)
    {
        try
        {
            var DoListBucketsAsyncResponse = await DoListBucketsAsync();
            context.Logger.LogInformation($"Results:
 {DoListBucketsAsyncResponse.Buckets}");

            context.Logger.LogInformation($"Successfully called ListBucketsAsync");
            return "Success!";
        }
        catch (Exception ex)
        {
            context.Logger.LogError($"Failed to call ListBucketsAsync: {ex.Message}");
```

```
            throw;
        }
    }

    private async Task<ListBucketsResponse> DoListBucketsAsync()
    {
        try
        {
            var putRequest = new ListBucketsRequest
            {
            };

            var response = await s3Client.ListBucketsAsync(putRequest);
            return response;
        }
        catch (AmazonS3Exception ex)
        {
            throw new Exception($"Failed to call ListBucketsAsync: {ex.Message}", ex);
        }
    }
}
```

Al invocar esta Lambda, verá la siguiente traza en el mapa de trazas de la consola: CloudWatch

Migrar a OpenTelemetry Python

Esta guía le ayuda a migrar las aplicaciones de Python del SDK de X-Ray a la OpenTelemetry instrumentación. Abarca los enfoques de instrumentación automática y manual, con ejemplos de código para escenarios comunes.

Secciones

- Soluciones de instrumentación automática sin código

- Instrumentación de las aplicaciones manualmente
- Inicialización de la configuración de seguimiento
- Seguimiento de solicitudes entrantes
- AWS Instrumentación del SDK
- Instrumentación de llamadas a HTTP salientes a través de solicitudes
- Compatibilidad de instrumentación para otras bibliotecas
- Creación manual de datos de rastros
- Instrumentación de Lambda

Soluciones de instrumentación automática sin código

Con el SDK de X-Ray, tenías que modificar el código de tu aplicación para rastrear las solicitudes. OpenTelemetry ofrece soluciones de autoinstrumentación sin código para rastrear las solicitudes. Con OpenTelemetry, tiene la opción de utilizar soluciones de autoinstrumentación sin código para rastrear las solicitudes.

Código cero con instrumentaciones automáticas basadas OpenTelemetry

1. Uso de la AWS distribución para la instrumentación automática OpenTelemetry (ADOT) para Python: para obtener información sobre la instrumentación automática para aplicaciones de Python, consulte Seguimiento y métricas con la distribución para la AWS instrumentación automática de Python. OpenTelemetry

   (Opcional) También puede habilitar CloudWatch Application Signals al instrumentar automáticamente sus aplicaciones AWS con la instrumentación automática de ADOT Python para monitorear el estado actual de las aplicaciones y realizar un seguimiento del rendimiento de las aplicaciones a largo plazo en comparación con sus objetivos comerciales. Application Signals le proporciona una visión unificada y centrada en las aplicaciones de sus aplicaciones, servicios y dependencias y lo ayuda a supervisar y evaluar el estado de las aplicaciones.
2. Uso de la instrumentación automática de código cero de OpenTelemetry Python: para la instrumentación automática con Python OpenTelemetry , consulte Instrumentación de código cero de Python.

Instrumentación de las aplicaciones manualmente

Puede instrumentar las aplicaciones manualmente mediante el comando `pip`.

With X-Ray SDK

```
pip install aws-xray-sdk
```

With OpenTelemetry SDK

```
pip install opentelemetry-api opentelemetry-sdk opentelemetry-exporter-otlp
 opentelemetry-propagator-aws-xray
```

Inicialización de la configuración de seguimiento

With X-Ray SDK

En X-Ray, el `xray_recorder` global se inicializa y se usa para generar segmentos y subsegmentos.

With OpenTelemetry SDK

> **Note**
>
> El muestreo remoto de X-Ray no está disponible actualmente para configurarlo para OpenTelemetry Python. Sin embargo, la compatibilidad para el muestreo remoto de X-Ray está disponible actualmente a través de la instrumentación automática de ADOT para Python.

En OpenTelemetry, necesita inicializar un global`TracerProvider`. Con este `TracerProvider`, puede adquirir un Rastreador que puede usar para generar intervalos en cualquier parte de la aplicación. Se recomienda que configure los siguientes componentes:

- `OTLPSpanExporter`— Necesario para exportar las trazas al CloudWatch OpenTelemetry agente/recopilador
- Un propagador de AWS rayos X: necesario para propagar el contexto de rastreo a AWS los servicios integrados con X-Ray

```
from opentelemetry import (
    trace,
    propagate
)
from opentelemetry.sdk.trace import TracerProvider

from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.exporter.otlp.proto.http.trace_exporter import OTLPSpanExporter
from opentelemetry.sdk.trace.export import BatchSpanProcessor
from opentelemetry.propagators.aws import AwsXRayPropagator

# Sends generated traces in the OTLP format to an OTel Collector running on port
 4318
otlp_exporter = OTLPSpanExporter(endpoint="http://localhost:4318/v1/traces")
# Processes traces in batches as opposed to immediately one after the other
span_processor = BatchSpanProcessor(otlp_exporter)
# More configurations can be done here. We will visit them later.

# Sets the global default tracer provider
provider = TracerProvider(active_span_processor=span_processor)
trace.set_tracer_provider(provider)

# Configures the global propagator to use the X-Ray Propagator
propagate.set_global_textmap(AwsXRayPropagator())

# Creates a tracer from the global tracer provider
tracer = trace.get_tracer("my.tracer.name")
# Use this tracer to create Spans
```

Con la instrumentación automática de ADOT para Python

Puede utilizar la instrumentación automática de ADOT para Python para configurar automáticamente OpenTelemetry sus aplicaciones de Python. Al utilizar la instrumentación automática de ADOT, no es necesario realizar cambios manuales en el código para rastrear las solicitudes entrantes ni para rastrear bibliotecas como el AWS SDK o los clientes HTTP. Para obtener más información, consulte Seguimiento y métricas con la AWS distribución para la instrumentación automática de OpenTelemetry Python.

La instrumentación automática de ADOT para Python admite:

- Muestreo remoto de X-Ray a través de la variable de entorno `export OTEL_TRACES_SAMPLER=xray`
- Propagación de contexto de rastros de X-Ray (habilitada de forma predeterminada)
- Detección de recursos (la detección de recursos para los entornos de Amazon EC2, Amazon ECS y Amazon EKS está habilitada de forma predeterminada)
- Las instrumentaciones de biblioteca automáticas para todas las OpenTelemetry instrumentaciones compatibles están habilitadas de forma predeterminada. Puede desactivarlas de forma selectiva a través de la variable de entorno `OTEL_PYTHON_DISABLED_INSTRUMENTATIONS` (todas están habilitadas de forma predeterminada)
- Creación manual de intervalos

Desde complementos del servicio de X-Ray hasta proveedores de OpenTelemetry AWS recursos

El SDK de X-Ray incluye complementos que se pueden añadir `xray_recorder` para capturar la información específica de la plataforma desde el servicio hospedado, como Amazon EC2, Amazon ECS y Elastic Beanstalk. Es similar a los proveedores de recursos en el sentido de OpenTelemetry que captura la información como atributos de recursos. Hay varios proveedores de recursos disponibles para diferentes AWS plataformas.

- Comience por instalar el paquete AWS de extensión, `pip install opentelemetry-sdk-extension-aws`
- Configure el detector de recursos deseado. El siguiente ejemplo muestra cómo configurar el proveedor de EC2 recursos de Amazon en el OpenTelemetry SDK.

```
from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.extension.aws.resource.ec2 import (
    AwsEc2ResourceDetector,
)
from opentelemetry.sdk.resources import get_aggregated_resources

provider = TracerProvider(
    active_span_processor=span_processor,
    resource=get_aggregated_resources([
        AwsEc2ResourceDetector(),
```

```
    ]))

trace.set_tracer_provider(provider)
```

Seguimiento de solicitudes entrantes

With X-Ray SDK

El X-Ray Python SDK admite marcos de aplicaciones como Django, Flask y Bottle para rastrear las solicitudes entrantes de las aplicaciones de Python que se ejecutan en ellos. Esto se hace agregando `XRayMiddleware` a la aplicación para cada marco.

With OpenTelemetry SDK

OpenTelemetry proporciona instrumentaciones para Django y Flask a través de las bibliotecas de instrumentación específicas. No hay instrumentación para Bottle disponible en OpenTelemetry, pero aún se pueden rastrear las aplicaciones utilizando la instrumentación WSGI. OpenTelemetry

Para el ejemplo de código siguiente, necesita la siguiente dependencia:

```
pip install opentelemetry-instrumentation-flask
```

Debe inicializar el OpenTelemetry SDK y registrar el global TracerProvider antes de añadir instrumentaciones al marco de su aplicación. Sin él, las operaciones de rastreo serán `no-ops`. Una vez que haya configurado el `TracerProvider` global, puede usar el instrumentor para el marco de la aplicación. El ejemplo siguiente muestra una aplicación de Flask.

```
from flask import Flask
from opentelemetry import trace
from opentelemetry.instrumentation.flask import FlaskInstrumentor
from opentelemetry.sdk.extension.aws.resource import AwsEc2ResourceDetector
from opentelemetry.sdk.resources import get_aggregated_resources
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.trace.export import BatchSpanProcessor, ConsoleSpanExporter

provider = TracerProvider(resource=get_aggregated_resources(
    [
```

```
        AwsEc2ResourceDetector(),
    ]))

processor = BatchSpanProcessor(ConsoleSpanExporter())
provider.add_span_processor(processor)
trace.set_tracer_provider(provider)

# Creates a tracer from the global tracer provider
tracer = trace.get_tracer("my.tracer.name")

app = Flask(__name__)

# Instrument the Flask app
FlaskInstrumentor().instrument_app(app)


@app.route('/')
def hello_world():
    return 'Hello World!'


if __name__ == '__main__':
    app.run()
```

AWS Instrumentación del SDK

With X-Ray SDK

El SDK de Python de X-Ray rastrea la solicitud del cliente del AWS SDK parcheando la `botocore` biblioteca. Para obtener más información, consulta Rastrear las llamadas AWS del SDK con el SDK de X-Ray para Python. En la aplicación, el método `patch_all()` se utiliza para instrumentar todas las bibliotecas o aplicar parches de forma selectiva mediante las bibliotecas de `botocore` o `boto3` que utilizan `patch((['botocore']))`. Cualquiera de los métodos elegidos instrumenta todos los clientes de Boto3 de la aplicación y genera un subsegmento para cualquier llamada realizada con estos clientes.

With OpenTelemetry SDK

Para el ejemplo de código siguiente, necesitará la siguiente dependencia:

```
pip install opentelemetry-instrumentation-botocore
```

Utilice la instrumentación de OpenTelemetry Botocore mediante programación para instrumentar todos los clientes de Boto3 de su aplicación. En el ejemplo siguiente se muestra la instrumentación de `botocore`.

```
import boto3
import opentelemetry.trace as trace
from botocore.exceptions import ClientError
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.resources import get_aggregated_resources
from opentelemetry.sdk.trace.export import (
    BatchSpanProcessor,
    ConsoleSpanExporter,
)
from opentelemetry.instrumentation.botocore import BotocoreInstrumentor

provider = TracerProvider()
processor = BatchSpanProcessor(ConsoleSpanExporter())
provider.add_span_processor(processor)
trace.set_tracer_provider(provider)

# Creates a tracer from the global tracer provider
tracer = trace.get_tracer("my.tracer.name")

# Instrument BotoCore
BotocoreInstrumentor().instrument()

# Initialize S3 client
s3 = boto3.client("s3", region_name="us-east-1")

# Your bucket name
bucket_name = "my-example-bucket"

# Get bucket location (as an example of describing it)
try:
    response = s3.get_bucket_location(Bucket=bucket_name)
    region = response.get("LocationConstraint") or "us-east-1"
    print(f"Bucket '{bucket_name}' is in region: {region}")

    # Optionally, get bucket's creation date via list_buckets
```

```
    buckets = s3.list_buckets()
    for bucket in buckets["Buckets"]:
        if bucket["Name"] == bucket_name:
            print(f"Bucket created on: {bucket['CreationDate']}")
            break
except ClientError as e:
    print(f"Failed to describe bucket: {e}")
```

Instrumentación de llamadas a HTTP salientes a través de solicitudes

With X-Ray SDK

El X-Ray Python SDK rastrea las llamadas a HTTP salientes a través de solicitudes parcheando la biblioteca de solicitudes. Para obtener más información, consulte Rastreo de llamadas a servicios web HTTP posteriores con el X-Ray SDK para Python. En la aplicación, puede usar el método `patch_all()` para instrumentar todas las bibliotecas o aplicar parches de forma selectiva a las bibliotecas de solicitudes mediante `patch(['requests'])`. Cualquiera de las opciones instrumenta la biblioteca de `requests` y genera un subsegmento para cualquier llamada realizada a través de `requests`.

With OpenTelemetry SDK

Para el ejemplo de código siguiente, necesitará la siguiente dependencia:

```
pip install opentelemetry-instrumentation-requests
```

Utilice la instrumentación de OpenTelemetry solicitudes mediante programación para instrumentar la biblioteca de solicitudes a fin de generar un seguimiento de las solicitudes HTTP que realice en su aplicación. Para obtener más información, consulta OpenTelemetry la sección Instrumentación de solicitudes. En el ejemplo siguiente se muestra la instrumentación de la biblioteca de `requests`.

```
from opentelemetry.instrumentation.requests import RequestsInstrumentor

# Instrument Requests
RequestsInstrumentor().instrument()
```

```
...

    example_session = requests.Session()
    example_session.get(url="https://example.com")
```

Otra opción, también puede instrumentar la biblioteca de `urllib3` subyacente para rastrear las solicitudes HTTP:

```
# pip install opentelemetry-instrumentation-urllib3
from opentelemetry.instrumentation.urllib3 import URLLib3Instrumentor

# Instrument urllib3
URLLib3Instrumentor().instrument()

...

    example_session = requests.Session()
    example_session.get(url="https://example.com")
```

Compatibilidad de instrumentación para otras bibliotecas

Puede encontrar la lista completa de instrumentaciones de biblioteca compatibles para OpenTelemetry Python en Bibliotecas, marcos, servidores de aplicaciones y. JVMs

También puedes buscar en el OpenTelemetry Registro si es OpenTelemetry compatible con la instrumentación. Consulte Registro para empezar a buscar.

Creación manual de datos de rastros

Puede crear segmentos y subsegmentos mediante `xray_recorder` en la aplicación de Python. Para obtener más información, consulte Instrumentación manual del código de Python. También puede agregar anotaciones y metadatos de forma manual a los datos de rastros.

Cómo crear intervalos con OpenTelemetry el SDK

Use la API de `start_as_current_span` para iniciar un intervalo y configúrelo para crear intervalos. Para ver ejemplos sobre la creación de intervalos, consulte Creación de intervalos. Una

vez que se inicia un intervalo y se encuentra en el ámbito actual, puede agregarle más información agregando atributos, eventos, excepciones, enlaces, etc. Al igual que tenemos segmentos y subsegmentos en X-Ray, hay diferentes tipos de intervalos. OpenTelemetry Solo los intervalos del tipo `SERVER` se convierten en segmentos de X-Ray, mientras que otros se convierten en subsegmentos de X-Ray.

```
from opentelemetry import trace
from opentelemetry.trace import SpanKind

import time

tracer = trace.get_tracer("my.tracer.name")

# Create a new span to track some work
with tracer.start_as_current_span("parent", kind=SpanKind.SERVER) as parent_span:
    time.sleep(1)

    # Create a nested span to track nested work
    with tracer.start_as_current_span("child", kind=SpanKind.CLIENT) as child_span:
        time.sleep(2)
        # the nested span is closed when it's out of scope

    # Now the parent span is the current span again
    time.sleep(1)

    # This span is also closed when it goes out of scope
```

Añadir anotaciones y metadatos a las trazas con el SDK OpenTelemetry

El SDK de Python de X-Ray proporciona anotaciones `put_annotation` y metadatos independientes APIs `put_metadata` para añadir anotaciones y metadatos a una traza. En el OpenTelemetry SDK, las anotaciones y los metadatos son simplemente atributos de un intervalo que se agregan a través de la `set_attribute` API.

Los atributos de intervalo que desee que sean anotaciones en un rastro se agregan en la clave reservada, `aws.xray.annotations` cuyo valor es una lista de pares de anotaciones clave-valor. Todos los demás atributos del intervalo se convierten en metadatos del segmento o subsegmento convertido.

Además, si utiliza el recopilador de ADOT, puede configurar qué atributos de intervalo deben convertirse en anotaciones de X-Ray especificando `indexed_attributes` en la configuración del recopilador.

En el siguiente ejemplo, se muestra cómo añadir anotaciones y metadatos a una traza mediante OpenTelemetry el SDK.

```
with tracer.start_as_current_span("parent", kind=SpanKind.SERVER) as parent_span:
    parent_span.set_attribute("TransactionId", "qwerty12345")
    parent_span.set_attribute("AccountId", "1234567890")

    # This will convert the TransactionId and AccountId to be searchable X-Ray
 annotations
    parent_span.set_attribute("aws.xray.annotations", ["TransactionId", "AccountId"])

    with tracer.start_as_current_span("child", kind=SpanKind.CLIENT) as child_span:

        # The MicroTransactionId will be converted to X-Ray metadata for the child
 subsegment
        child_span.set_attribute("MicroTransactionId", "micro12345")
```

Instrumentación de Lambda

Para supervisar las funciones de lambda en X-Ray, habilite X-Ray y agregue los permisos adecuados al rol de invocación de la función. Además, si está rastreando las solicitudes posteriores de la función, estaría instrumentando el código con X-Ray Python SDK.

En OpenTelemetry el caso de X-Ray, se recomienda utilizar la capa lambda de CloudWatch Application Signals con Application Signals desactivada. Esto instrumentará automáticamente la función y generará intervalos para la invocación de la función y cualquier solicitud posterior de la función. Además del rastreo, si está interesado en usar señales de las aplicaciones para supervisar el estado de su función, consulte Habilitar las aplicaciones en Lambda.

- Busque el ARN de la capa Lambda necesario para su función en AWS Lambda Layer for y agréguelo. OpenTelemetry ARNs
- Establezca las siguientes variables de entorno para la función.
  - `AWS_LAMBDA_EXEC_WRAPPER=/opt/otel-instrument`: esto carga la instrumentación automática de la función

- `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`: esto desactivará la supervisión de las señales de las aplicaciones

Creación manual de intervalos con instrumentación de Lambda

Además, puede generar intervalos personalizados dentro de la función para realizar un seguimiento del trabajo. Puede hacerlo utilizando solo el paquete `opentelemetry-api` junto con la instrumentación automática de la capa de Lambda de señales de las aplicaciones.

1. Incluir la `opentelemetry-api` como una dependencia en la función
2. El siguiente fragmento de código es un ejemplo para generar intervalos personalizados

```
from opentelemetry import trace

# Get the tracer (auto-configured by the Application Signals layer)
tracer = trace.get_tracer(__name__)

def handler(event, context):
    # This span is a child of the layer's root span
    with tracer.start_as_current_span("my-custom-span") as span:
        span.set_attribute("key1", "value1")
        span.add_event("custom-event", {"detail": "something happened"})

        # Any logic you want to trace
        result = some_internal_logic()

    return {
        "statusCode": 200,
        "body": result
    }
```

Migre a Ruby OpenTelemetry

Para migrar tus aplicaciones de Ruby del SDK de X-Ray a la OpenTelemetry instrumentación, usa los siguientes ejemplos de código y guías para la instrumentación manual.

Secciones

Instrumentación manual de las soluciones con el SDK

Tracing setup with X-Ray SDK

X-Ray SDK para Ruby requería que configurara el código con complementos de servicio.

```
require 'aws-xray-sdk'

XRay.recorder.configure(plugins: [:ec2, :elastic_beanstalk])
```

Tracing setup with OpenTelemetry SDK

> **Note**
>
> El muestreo remoto de X-Ray actualmente no está disponible para configurarlo para OpenTelemetry Ruby.

Para una aplicación de Ruby on Rails, coloque el código de configuración en un inicializador de Rails. Para obtener más información, consulte Introducción. Para todos los programas de Ruby instrumentados manualmente, debes usar el `OpenTelemetry::SDK.configure` método para configurar el SDK de OpenTelemetry Ruby.

Primero, instale los siguientes paquetes:

```
bundle add opentelemetry-sdk opentelemetry-exporter-otlp opentelemetry-propagator-
xray
```

A continuación, configure el OpenTelemetry SDK mediante el código de configuración que se ejecuta cuando se inicializa el programa. Se recomienda que configure los siguientes componentes:

- `OTLP Exporter`— Necesario para exportar las trazas al CloudWatch agente y OpenTelemetry al recopilador
- `An AWS X-Ray Propagator`— Necesario para propagar el contexto de rastreo a AWS los servicios que están integrados con X-Ray

```
require 'opentelemetry-sdk'
require 'opentelemetry-exporter-otlp'

# Import the gem containing the AWS X-Ray for OTel Ruby ID Generator and propagator
require 'opentelemetry-propagator-xray'

OpenTelemetry::SDK.configure do |c|
  c.service_name = 'my-service-name'

  c.add_span_processor(
    # Use the BatchSpanProcessor to send traces in groups instead of one at a time
    OpenTelemetry::SDK::Trace::Export::BatchSpanProcessor.new(
      # Use the default OLTP Exporter to send traces to the ADOT Collector
      OpenTelemetry::Exporter::OTLP::Exporter.new(
        # The OpenTelemetry Collector is running as a sidecar and listening on port
 4318
        endpoint:"http://127.0.0.1:4318/v1/traces"
      )
    )
  )

  # The X-Ray Propagator injects the X-Ray Tracing Header into downstream calls
  c.propagators = [OpenTelemetry::Propagator::XRay::TextMapPropagator.new]
end
```

OpenTelemetry SDKs también tienen el concepto de instrumentaciones de biblioteca. Al habilitarlas, se crearán automáticamente intervalos para bibliotecas como el SDK. AWS OpenTelemetry ofrece la opción de habilitar todas las instrumentaciones de la biblioteca o especificar qué instrumentaciones de la biblioteca se deben habilitar.

Para habilitar todas las instrumentaciones, primero instale el paquete `opentelemetry-instrumentation-all`:

```
bundle add opentelemetry-instrumentation-all
```

A continuación, actualice la configuración para habilitar todas las instrumentaciones de la biblioteca, como se muestra a continuación:

```
require 'opentelemetry/instrumentation/all'
...

OpenTelemetry::SDK.configure do |c|
   ...

  c.use_all() # Enable all instrumentations
end
```

OpenTelemetry SDKs también incluyen el concepto de instrumentaciones de biblioteca. Al habilitarlas, se crearán automáticamente intervalos para bibliotecas como el SDK. AWS OpenTelemetry ofrece la opción de habilitar todas las instrumentaciones de la biblioteca o especificar qué instrumentaciones de la biblioteca se deben habilitar.

Para habilitar todas las instrumentaciones, primero instale el paquete `opentelemetry-instrumentation-all`:

```
bundle add opentelemetry-instrumentation-all
```

A continuación, actualice la configuración para habilitar todas las instrumentaciones de la biblioteca, como se muestra a continuación:

```
require 'opentelemetry/instrumentation/all'
...

OpenTelemetry::SDK.configure do |c|
   ...

  c.use_all() # Enable all instrumentations
```

```
end
```

Rastreo de las solicitudes entrantes (instrumentación de Rails)

With X-Ray SDK

Con el X-Ray SDK, el rastreo de X-Ray se configura para el marco de Rails tras la inicialización.

Ejemploconfig/initializers/aws: _xray.rb

```
Rails.application.config.xray = {
  name: 'my app',
  patch: %I[net_http aws_sdk],
  active_record: true
}
```

With OpenTelemetry SDK

Primero, instale los siguientes paquetes:

```
bundle add opentelemetry-instrumentation-rack opentelemetry-instrumentation-
rails opentelemetry-instrumentation-action_pack opentelemetry-instrumentation-
active_record opentelemetry-instrumentation-action_view
```

A continuación, actualice la configuración para habilitar la instrumentación de la aplicación de Rails, como se muestra a continuación:

```
# During SDK configuration
OpenTelemetry::SDK.configure do |c|

  ...

  c.use 'OpenTelemetry::Instrumentation::Rails'
  c.use 'OpenTelemetry::Instrumentation::Rack'
  c.use 'OpenTelemetry::Instrumentation::ActionPack'
  c.use 'OpenTelemetry::Instrumentation::ActiveSupport'
  c.use 'OpenTelemetry::Instrumentation::ActionView'
```

```
  ...

end
```

AWS Instrumentación del SDK

With X-Ray SDK

Para instrumentar AWS las solicitudes salientes del AWS SDK, los clientes del AWS SDK se parchean con X-Ray, como en el siguiente ejemplo:

```
require 'aws-xray-sdk'
require 'aws-sdk-s3'

# Patch AWS SDK clients
XRay.recorder.configure(plugins: [:aws_sdk])

# Use the instrumented client
s3 = Aws::S3::Client.new
s3.list_buckets
```

With OpenTelemetry SDK

AWS El SDK for Ruby V3 admite el registro y la emisión OpenTelemetry de trazas. Para obtener información sobre cómo configurar OpenTelemetry un cliente de servicio, consulta Cómo configurar las funciones de observabilidad en el AWS SDK for Ruby.

Instrumentación de llamadas a HTTP salientes

Al realizar llamadas a HTTP a servicios externos, es posible que tenga que instrumentarlas manualmente si la instrumentación automática no está disponible o no proporciona suficientes detalles.

With X-Ray SDK

Para instrumentar llamadas posteriores, se usó el X-Ray SDK para Ruby con el fin de aplicar parches a la biblioteca de `net/http` que utiliza la aplicación:

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http]
}

XRay.recorder.configure(config)
```

With OpenTelemetry SDK

Para habilitar el uso de la `net/http` instrumentación OpenTelemetry, primero instale el `opentelemetry-instrumentation-net_http` paquete:

```
bundle add opentelemetry-instrumentation-net_http
```

A continuación, actualice la configuración para habilitar la instrumentación de `net/http`, como se muestra a continuación:

```
OpenTelemetry::SDK.configure do |c|
   ...

  c.use 'OpenTelemetry::Instrumentation::Net::HTTP'
  ...

end
```

Compatibilidad de instrumentación para otras bibliotecas

Puede encontrar la lista completa de instrumentaciones de biblioteca compatibles con Ruby en. OpenTelemetry opentelemetry-ruby-contrib

Como alternativa, puedes buscar en el OpenTelemetry Registro para averiguar si es OpenTelemetry compatible con la instrumentación. Para obtener más información, consulte Registro.

Creación manual de datos de rastros

With X-Ray SDK

Con X-Ray, el paquete de `aws-xray-sdk` que requería para crear segmentos manualmente y sus subsegmentos secundarios a fin de rastrear la aplicación. Es posible que también haya agregado anotaciones y metadatos de X-Ray a los segmentos o subsegmentos:

```
require 'aws-xray-sdk'
...

# Start a segment
segment = XRay.recorder.begin_segment('my-service')

# Add annotations (indexed key-value pairs)
segment.annotations[:user_id] = 'user-123'
segment.annotations[:payment_status] = 'completed'

# Add metadata (non-indexed data)
segment.metadata[:order] = {
  id: 'order-456',
  items: [
    { product_id: 'prod-1', quantity: 2 },
    { product_id: 'prod-2', quantity: 1 }
  ],
  total: 67.99
}

# Add metadata to a specific namespace
segment.metadata(namespace: 'payment') do |metadata|
  metadata[:transaction_id] = 'tx-789'
  metadata[:payment_method] = 'credit_card'
end

# Create a subsegment with annotations and metadata
segment.subsegment('payment-processing') do |subsegment1|
  subsegment1.annotations[:payment_id] = 'pay-123'
  subsegment1.metadata[:details] = { amount: 67.99, currency: 'USD' }

  # Create a nested subsegment
  subsegment1.subsegment('operation-2') do |subsegment2|
    # Do more work...
  end
```

```
end

# Close the segment
segment.close
```

With OpenTelemetry SDK

Puede utilizar intervalos personalizados para supervisar el rendimiento de las actividades internas que no se recopilan en las bibliotecas de instrumentación. Tenga en cuenta que solo los intervalos de tipo servidor se convierten en segmentos de X-Ray, todos los demás intervalos se convierten en subsegmentos de X-Ray. Por defecto, los intervalos son `INTERNAL`.

En primer lugar, cree un rastreador para generar intervalos, que puede obtener mediante el método `OpenTelemetry.tracer_provider.tracer('<YOUR_TRACER_NAME>')`. Esto proporcionará una instancia de Tracer que se registrará globalmente en la OpenTelemetry configuración de su aplicación. Es común tener un solo rastreador para toda la aplicación. Cree un OpenTelemetry rastreador y utilícelo para crear intervalos:

```
require 'opentelemetry-sdk'

...

# Get a tracer
tracer = OpenTelemetry.tracer_provider.tracer('my-application')

# Create a server span (equivalent to X-Ray segment)
tracer.in_span('my-application', kind: OpenTelemetry::Trace::SpanKind::SERVER) do |
span|
  # Do work...

  # Create nested spans of default kind INTERNAL will become an X-Ray subsegment
  tracer.in_span('operation-1') do |child_span1|
    # Set attributes (equivalent to X-Ray annotations and metadata)
    child_span1.set_attribute('key', 'value')

    # Do more work...
    tracer.in_span('operation-2') do |child_span2|
      # Do more work...
    end
  end
end
```

Añadir anotaciones y metadatos a las trazas con el SDK OpenTelemetry

Use el método `set_attribute` para agregar atributos a cada intervalo. Tenga en cuenta que de forma predeterminada, todos estos atributos de intervalo se convertirán en metadatos en los datos sin procesar de X-Ray. Para garantizar que un atributo se convierta en una anotación y no en metadatos, puede agregar esa clave del atributo a la lista del atributo `aws.xray.annotations`. Para obtener más información, consulte Habilitar anotaciones de X-Ray personalizadas.

```
# SERVER span will become an X-Ray segment
tracer.in_span('my-server-operation', kind: OpenTelemetry::Trace::SpanKind::SERVER)
 do |span|
    # Your server logic here
    span.set_attribute('attribute.key', 'attribute.value')
    span.set_attribute("metadataKey", "metadataValue")
    span.set_attribute("annotationKey1", "annotationValue")

    # Create X-Ray annotations
    span.set_attribute("aws.xray.annotations", ["annotationKey1"])
end
```

Instrumentación manual de Lambda

With X-Ray SDK

Después de que se habilitase el rastreo activo en Lambda, no se requiere ninguna configuración adicional para usar el X-Ray SDK. Lambda creará un segmento que represente la invocación del controlador de Lambda y puede crear subsegmentos o bibliotecas de instrumentos mediante el X-Ray SDK sin necesidad de realizar ninguna configuración adicional.

With OpenTelemetry SDK

Tenga en cuenta el código de la función de Lambda de ejemplo siguiente (sin instrumentación):

```
require 'json'
def lambda_handler(event:, context:)
    # TODO implement
    { statusCode: 200, body: JSON.generate('Hello from Lambda!') }
```

```
end
```

Para instrumentar su Lambda de forma manual, necesitará lo siguiente:

1. Agregue las siguientes gemas para su Lambda

   ```
   gem 'opentelemetry-sdk'
   gem 'opentelemetry-exporter-otlp'
   gem 'opentelemetry-propagator-xray'
   gem 'aws-distro-opentelemetry-exporter-xray-udp'
   gem 'opentelemetry-instrumentation-aws_lambda'
   gem 'opentelemetry-propagator-xray', '~> 0.24.0' # Requires version v0.24.0 or
    higher
   ```

2. Inicialice el OpenTelemetry SDK fuera de su controlador Lambda. Se recomienda configurar el OpenTelemetry SDK con:

   1. Un procesador de intervalos simple con un exportador de intervalos de X-Ray UDP para enviar rastros al punto de conexión de UDP X-Ray de Lambda
   2. Un propagador de X-Ray Lambda
   3. Configuración de `service_name` que se establecerá en el nombre de la función de Lambda

3. En la clase del controlador de Lambda, agregue las siguientes líneas para instrumentar el controlador de Lambda:

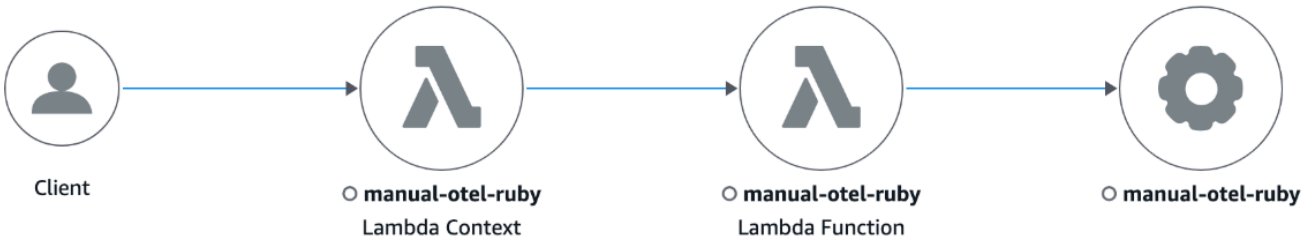
   ```
   class Handler
         extend OpenTelemetry::Instrumentation::AwsLambda::Wrap
         ...

         instrument_handler :process
     end
   ```

El código siguiente muestra la función de Lambda después de los cambios necesarios. Puede crear intervalos personalizados adicionales para complementar los intervalos que se proporcionan automáticamente.

```
require 'json'
```

```
require 'opentelemetry-sdk'
require 'aws/distro/opentelemetry/exporter/xray/udp'
require 'opentelemetry/propagator/xray'
require 'opentelemetry/instrumentation/aws_lambda'

# Initialize OpenTelemetry SDK outside handler
OpenTelemetry::SDK.configure do |c|
  # Configure the AWS Distro for OpenTelemetry X-Ray Lambda exporter
  c.add_span_processor(
    OpenTelemetry::SDK::Trace::Export::SimpleSpanProcessor.new(
      AWS::Distro::OpenTelemetry::Exporter::XRay::UDP::AWSXRayUDPSpanExporter.new
    )
  )

  # Configure X-Ray Lambda propagator
  c.propagators = [OpenTelemetry::Propagator::XRay.lambda_text_map_propagator]

  # Set minimal resource information
  c.resource = OpenTelemetry::SDK::Resources::Resource.create({
    OpenTelemetry::SemanticConventions::Resource::SERVICE_NAME =>
 ENV['AWS_LAMBDA_FUNCTION_NAME']
  })
  c.use 'OpenTelemetry::Instrumentation::AwsLambda'
end

module LambdaFunctions
  class Handler
    extend OpenTelemetry::Instrumentation::AwsLambda::Wrap
    def self.process(event:, context:)
      "Hello!"
    end
    instrument_handler :process
  end
end
```

El siguiente es un ejemplo de mapa de rastros de una función de Lambda instrumentada escrita en Ruby.



También puede usar capas Lambda OpenTelemetry para configurar su Lambda. Para obtener más información, consulte OpenTelemetry AWS-Lambda Instrumentation.

Creación de recursos de X-Ray con AWS CloudFormation

AWS X-Ray está integrado con AWS CloudFormation, un servicio que le ayuda a modelar y configurar sus recursos de AWS para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Puede crear una plantilla que describa todos los recursos de AWS que desee, y CloudFormation aprovisionará y configurará estos recursos por usted.

Cuando utiliza CloudFormation, puede volver a utilizar la plantilla para configurar sus recursos de X-Ray de forma coherente y repetida. Solo tiene que describir los recursos una vez y luego aprovisionar los mismos recursos una y otra vez en varias Cuentas de AWS y regiones.

X-Ray y plantillas de CloudFormation

Para aprovisionar y configurar los recursos de X-Ray y sus servicios conexos, debe entender las plantillas de CloudFormation. Las plantillas son archivos de texto con formato JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus pilas de CloudFormation. Si no está familiarizado con JSON o YAML, puede utilizar Designer de CloudFormation para comenzar a utilizar las plantillas de CloudFormation. Para obtener más información, consulte ¿Qué es Designer de CloudFormation? en la Guía del usuario de AWS CloudFormation.

X-Ray admite la creación de recursos `AWS::XRay::Group`, `AWS::XRay::SamplingRule` y `AWS::XRay::ResourcePolicy` en CloudFormation. Para obtener más información junto con ejemplos de plantillas JSON y YAML, consulte la referencia de tipos de recurso de X-Ray en la Guía del usuario de AWS CloudFormation.

Obtención de más información sobre CloudFormation

Para conocer más información acerca de CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- AWS CloudFormation Guía del usuario de
- CloudFormation Referencia de la API de
- AWS CloudFormation Guía del usuario de la interfaz de la línea de comandos de

Etiquetado de reglas de muestreo y grupos de X-Ray

Las etiquetas son palabras o frases que puede utilizar para identificar y organizar sus recursos de AWS. Puede agregar varias etiquetas a cada recurso. Cada etiqueta incluye una clave y un valor opcional que define el usuario. Por ejemplo, la clave de etiqueta podría ser **`domain`** y el valor de etiqueta podría ser **`example.com`**. Puede buscar y filtrar sus recursos en función de las etiquetas que añada. Para más información sobre el uso de etiquetas, consulte Etiquetado de recursos de AWS en la Guía de referencia general de AWS.

Puede utilizar etiquetas para aplicar permisos basados en etiquetas a las distribuciones de CloudFront. Para obtener más información, consulte Control de acceso a los recursos de AWS mediante etiquetas.

> **Note**
>
> Actualmente, ni Tag Editor ni AWS Resource Groups admiten recursos de X-Ray. Las etiquetas se agregan y administran mediante la consola o la API de AWS X-Ray.

También puede aplicar etiquetas a recursos a través de la consola, la API y los SDK de X-Ray, la AWS CLI y AWS Tools for Windows PowerShell. Para obtener más información, consulte la documentación siguiente:

- API de X-Ray: consulte las siguientes operaciones en la referencia de la API de AWS X-Ray:
  - ListTagsForResource
  - CreateSamplingRule
  - CreateGroup
  - TagResource
  - UntagResource
- AWS CLI: consulte xray en la Referencia de comandos de la AWS CLI.
- SDK: consulte la documentación del SDK correspondiente en la página Documentación de AWS.

> **Note**
>
> Si no puede agregar o cambiar etiquetas en un recurso de X-Ray, o no puede agregar un recurso que tenga etiquetas específicas, es posible que no tenga permisos para realizar

esta operación. Para solicitar acceso, póngase en contacto con un usuario de AWS de su empresa que tenga permisos de administrador en X-Ray.

Temas

Restricciones de las etiquetas

Se aplican las siguientes restricciones a las etiquetas.

- Número máximo de etiquetas por recurso: 50
- Longitud máxima de la clave: 128 caracteres Unicode.
- Longitud máxima del valor: 256 caracteres Unicode.
- Valores válidos para claves y valores: a-z, A-Z, 0-9, espacio y los siguientes caracteres: _ . : / = + - y @
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas.
- No utilice `aws:` como prefijo para claves, ya que está reservado para AWS.

> **Note**
>
> No puede editar ni eliminar las etiquetas del sistema.

Administración de etiquetas en la consola

Puede añadir etiquetas opcionales al crear un grupo de X-Ray o una regla de muestreo. Las etiquetas también se pueden cambiar o eliminar en la consola más adelante.

En los siguientes procedimientos se explica cómo agregar, editar y eliminar etiquetas para sus grupos y reglas de muestreo en la consola X-Ray.

Temas

Agregar etiquetas a un nuevo grupo (consola)

Al crear un nuevo grupo de X-Ray, puede añadir etiquetas opcionales en la página Crear grupo.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. En el panel de navegación, expanda Configuración y elija Grupos.
3. Elija Crear grupo.
4. En la página Crear grupo, especifique un nombre y una expresión de filtro para el grupo. Para obtener más información sobre estas propiedades, consulte Configuración de grupos.
5. En Etiquetas, introduzca una clave de etiqueta y, si lo desea, un valor de etiqueta. Por ejemplo, puede introducir una clave de etiqueta de **`Stage`** y un valor de etiqueta de **`Production`** para indicar que este grupo es para uso en producción. Al añadir una etiqueta, aparece una nueva línea para que añada otra etiqueta, si es necesario. Consulte Restricciones de las etiquetas en este tema para conocer las limitaciones de las etiquetas.
6. Cuando termine de agregar etiquetas, elija Crear grupo.

Agregar etiquetas a una nueva regla de muestreo (consola)

Al crear una nueva regla de muestreo de X-Ray, puede añadir etiquetas en la página Crear regla de muestreo.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. En el panel de navegación, expanda Configuración y elija Muestreo.
3. Seleccione Crear regla de muestreo.

4. En la página Crear regla de muestreo, especifique el nombre, la prioridad, los límites, los criterios coincidentes y los atributos coincidentes. Para obtener más información sobre estas propiedades, consulte Configuración de reglas de muestreo de.
5. En Etiquetas, introduzca una clave de etiqueta y, si lo desea, un valor de etiqueta. Por ejemplo, puede introducir una clave de etiqueta de **`Stage`** y un valor de etiqueta de **`Production`** para indicar que esta regla de muestreo es para uso en producción. Al añadir una etiqueta, aparece una nueva línea para que añada otra etiqueta, si es necesario. Consulte Restricciones de las etiquetas en este tema para conocer las limitaciones de las etiquetas.
6. Cuando termine de agregar etiquetas, elija Crear regla de muestreo.

Edición o eliminación de etiquetas de un grupo (consola)

Puede cambiar o eliminar etiquetas de un grupo de X-Ray en la página Editar grupo.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. En el panel de navegación, expanda Configuración y elija Grupos.
3. En la tabla Grupos, elija el nombre del un grupo.
4. En la página Editar grupo, en Etiquetas, edite las claves y los valores de las etiquetas. No puede tener claves de etiquetas duplicadas. Los valores de las etiquetas son opcionales; puede eliminarlos si lo desea. Para obtener más información sobre otras propiedades de la página Editar grupo, consulte Configuración de grupos. Consulte Restricciones de las etiquetas en este tema para conocer las limitaciones de las etiquetas.
5. Para eliminar una etiqueta, elija X a la derecha de la etiqueta.
6. Cuando haya terminado de editar o eliminar etiquetas, elija Actualizar grupo.

Edición o eliminación de etiquetas de una regla de muestreo (consola)

Puede cambiar o eliminar las etiquetas de una regla de muestreo de X-Ray en la página Editar regla de muestreo.

1. Inicie sesión en la Consola de administración de AWS y abra la consola de X-Ray en https://console.aws.amazon.com/xray/home.
2. En el panel de navegación, expanda Configuración y elija Muestreo.
3. En la tabla Reglas de muestreo, elija el nombre de una regla de muestreo.

4. En Etiquetas, edite las claves y los valores de las etiquetas. No puede tener claves de etiquetas duplicadas. Los valores de las etiquetas son opcionales; puede eliminarlos si lo desea. Para obtener más información sobre otras propiedades de la página Editar regla de muestreo, consulte Configuración de reglas de muestreo de. Consulte Restricciones de las etiquetas en este tema para conocer las limitaciones de las etiquetas.
5. Para eliminar una etiqueta, elija X a la derecha de la etiqueta.
6. Cuando haya terminado de editar o eliminar etiquetas, elija Actualizar regla de muestreo.

Administración de etiquetas en la AWS CLI

Puede añadir etiquetas al crear un grupo o una regla de muestreo de X-Ray. También puede utilizar la AWS CLI para crear y administrar etiquetas. Para actualizar las etiquetas de un grupo o de una regla de muestreo existente, use la consola de AWS X-Ray o las API TagResource o UntagResource.

Temas

- Agregar etiquetas a un nuevo grupo de o regla de muestreo de X-Ray (CLI)
- Añadir etiquetas a un recurso existente (CLI)
- Visualizar las etiquetas de un recurso (CLI)
- Eliminar etiquetas de un recurso (CLI)

Agregar etiquetas a un nuevo grupo de o regla de muestreo de X-Ray (CLI)

Para añadir etiquetas opcionales al crear un nuevo grupo o una nueva regla de muestreo de X-Ray, utilice uno de los siguientes comandos.

- Para añadir etiquetas a un nuevo grupo, ejecute el siguiente comando y sustituya *group_name* por el nombre de su grupo, *mydomain.com* por el punto de conexión de su servicio, *key_name* por una clave de etiqueta y, si lo desea, *value* por un valor de etiqueta. Para obtener más información sobre cómo crear un grupo, consulte Grupos.

```
aws xray create-group \
   --group-name "group_name" \
   --filter-expression "service(\"mydomain.com\") {fault OR error}" \
   --tags [{"Key": "key_name","Value": "value"},{"Key": "key_name","Value": "value"}]
```

A continuación se muestra un ejemplo.

```
aws xray create-group \
   --group-name "AdminGroup" \
   --filter-expression "service(\"mydomain.com\") {fault OR error}" \
   --tags [{"Key": "Stage","Value": "Prod"},{"Key": "Department","Value": "QA"}]
```

- Para añadir etiquetas a una nueva regla de muestreo, ejecute el siguiente comando y sustituya *key_name* por una clave de etiqueta y, si lo desea, *value* por un valor de etiqueta. Este comando especifica los valores del parámetro `--sampling-rule` como archivo JSON. Para obtener más información sobre cómo crear una regla de muestreo, consulte Reglas de muestreo.

```
aws xray create-sampling-rule \
   --cli-input-json file://file_name.json
```

A continuación puede ver el contenido del archivo JSON *file_name.json* especificado por el parámetro `--cli-input-json`.

```
{
    "SamplingRule": {
        "RuleName": "rule_name",
        "RuleARN": "string",
        "ResourceARN": "string",
        "Priority": integer,
        "FixedRate": double,
        "ReservoirSize": integer,
        "ServiceName": "string",
        "ServiceType": "string",
        "Host": "string",
        "HTTPMethod": "string",
        "URLPath": "string",
        "Version": integer,
        "Attributes": {"attribute_name": "value","attribute_name": "value"...}
    }
    "Tags": [
           {
               "Key":"key_name",
               "Value":"value"
           },
           {
               "Key":"key_name",
```

```
                "Value":"value"
            }
           ]
}
```

El siguiente comando es un ejemplo.

```
aws xray create-sampling-rule \
   --cli-input-json file://9000-base-scorekeep.json
```

A continuación puede ver el contenido del archivo de ejemplo `9000-base-scorekeep.json` especificado por el parámetro `--cli-input-json`.

```
{
    "SamplingRule": {
        "RuleName": "base-scorekeep",
        "ResourceARN": "*",
        "Priority": 9000,
        "FixedRate": 0.1,
        "ReservoirSize": 5,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1
    }
    "Tags": [
            {
                "Key":"Stage",
                "Value":"Prod"
            },
            {
                "Key":"Department",
                "Value":"QA"
            }
           ]
}
```

Añadir etiquetas a un recurso existente (CLI)

Puede ejecutar el comando `tag-resource` para añadir etiquetas a un grupo o una regla de muestreo de X-Ray existente. Este método puede resultar más sencillo que añadir etiquetas ejecutando `update-group` o `update-sampling-rule`.

Para agregar etiquetas a un grupo o a una regla de muestreo, ejecute el siguiente comando, sustituyendo el ARN por el ARN del recurso y especificando las claves y los valores opcionales de las etiquetas que desee agregar.

```
aws xray tag-resource \
   --resource-arn "ARN" \
   --tag-keys [{"Key":"key_name","Value":"value"}, {"Key":"key_name","Value":"value"}]
```

A continuación se muestra un ejemplo.

```
aws xray tag-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup" \
   --tag-keys [{"Key": "Stage","Value": "Prod"},{"Key": "Department","Value": "QA"}]
```

Visualizar las etiquetas de un recurso (CLI)

Puede ejecutar el comando `list-tags-for-resource` para ver una lista de las etiquetas de un grupo o una regla de muestreo de X-Ray.

Para ver una lista de las etiquetas asociadas a un grupo o una regla de muestreo, ejecute el siguiente comando sustituyendo el ARN por el ARN del recurso.

```
aws xray list-tags-for-resource \
   --resource-arn "ARN"
```

A continuación se muestra un ejemplo.

```
aws xray list-tags-for-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup"
```

Eliminar etiquetas de un recurso (CLI)

Puede ejecutar el comando `untag-resource` para eliminar las etiquetas de un grupo o una regla de muestreo de X-Ray.

Para agregar etiquetas de un grupo o de una regla de muestreo, ejecute el siguiente comando, sustituyendo el ARN por el ARN del recurso y especificando las claves de las etiquetas que desee eliminar.

Con el comando `untag-resource` solo puede eliminar etiquetas enteras. Para eliminar los valores de las etiquetas, utilice la consola de X-Ray o elimine las etiquetas y añada otras nuevas con las mismas claves pero con valores diferentes o vacíos.

```
aws xray untag-resource \
   --resource-arn "ARN" \
   --tag-keys ["key_name","key_name"]
```

A continuación se muestra un ejemplo.

```
aws xray untag-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/group_name" \
   --tag-keys ["Stage","Department"]
```

Controlar el acceso a los recursos de X-Ray en función de etiquetas

Puede adjuntar etiquetas a los grupos o las reglas de muestreo de X-Ray, o pasar las etiquetas en una solicitud a X-Ray. Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `xray:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Para obtener más información sobre estas claves de condición, consulte Control de acceso a los recursos de AWS mediante etiquetas.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Gestión del acceso a los grupos de rayos X y a las reglas de muestreo en función de las etiquetas.

Resolución de problemas de AWS X-Ray

En este tema se enumeran errores y problemas comunes que podrían surgir cuando se utiliza la consola, la API o los SDK de X-Ray. Si se encuentra con un problema que no aparezca en esta lista, puede utilizar el botón Comentarios de esta página para notificarlo.

Secciones

Mapa de rastros de X-Ray y páginas de datos de rastro

Las siguientes secciones pueden ayudarle si tiene problemas para usar el mapa de rastros de X-Ray y la página de detalles del rastro:

No veo todos mis registros de CloudWatch

La forma de configurar los registros para que aparezcan en las páginas de datos de rastro y del mapa de rastros de X-Ray depende de cada servicio.

- Los registros de API Gateway aparecen si se activa el registro en API Gateway.

No todos los nodos del mapa de servicio permiten ver los registros asociados. Consulte los registros de los siguientes tipos de nodo:

- Contexto de Lambda
- Función de Lambda
- Etapa de API Gateway
- Clúster de Amazon ECS
- Instancia de Amazon ECS
- Servicio de Amazon ECS
- Tarea de Amazon ECS

- Clúster de Amazon EKS
- Espacio de nombres de Amazon EKS
- Nodo de Amazon EKS
- Pod de Amazon EKS
- Servicio de Amazon EKS

No veo todas mis alarmas en el mapa de rastros de X-Ray

El mapa de rastros de X-Ray muestra solo el icono de alerta de un nodo si cualquier alarma asociada a ese nodo tiene el estado ALARM.

El mapa de rastros asocia alarmas con nodos mediante la siguiente lógica:

- Si el nodo representa un servicio de AWS, todas las alarmas con el espacio de nombres asociado a ese servicio se asocian al nodo. Por ejemplo, un nodo de tipo `AWS::Kinesis` está vinculado a todas las alarmas basadas en métricas en el espacio de nombres de CloudWatch `AWS/Kinesis`.
- Si el nodo representa un recurso de AWS, las alarmas de ese recurso específico están vinculadas. Por ejemplo, un nodo de tipo `AWS::DynamoDB::Table` con el nombre “MyTable” está vinculado a todas las alarmas basadas en una métrica con el espacio de nombres `AWS/DynamoDB` y que tienen la dimensión `TableName` establecida en `MyTable`.
- Si el nodo es de tipo desconocido, que se identifica mediante un borde discontinuo alrededor del nombre, no se asociará ninguna alarma a ese nodo.

No veo algunos de los recursos de AWS en el mapa de rastros

No todos los recursos de AWS se representan mediante un nodo dedicado. Algunos servicios de AWS se representan mediante un solo nodo para todas las solicitudes al servicio. Los siguientes tipos de recurso se muestran con un nodo por recurso:

- `AWS::DynamoDB::Table`
- `AWS::Lambda::Function`

  Las funciones de Lambda se representan mediante dos nodos: uno para el contenedor Lambda y otro para la función. Esto ayuda a identificar problemas de arranque en frío con las funciones de Lambda. Los nodos de contenedor de Lambda se asocian a alarmas y paneles de la misma manera que los nodos de función de Lambda.

- `AWS::ApiGateway::Stage`
- `AWS::SQS::Queue`
- `AWS::SNS::Topic`

Hay demasiados nodos en el mapa de rastros

Utilice grupos de X-Ray para dividir el mapa en varios mapas. Para obtener más información, consulte Uso de expresiones de filtro con grupos.

SDK de X-Ray para Java

Error: Exception in thread "Thread-1" com.amazonaws.xray.exceptions.SegmentNotFoundException: Failed to begin subsegment named 'AmazonSNS': segment cannot be found.

Este error indica que el SDK de X-Ray ha intentado registrar una llamada saliente a AWS, pero no ha podido encontrar un segmento abierto. Esto podría darse en las siguientes situaciones:

- No se ha configurado un filtro de servlet: el SDK de X-Ray crea segmentos para las solicitudes entrantes con un filtro denominado `AWSXRayServletFilter`. Configure un filtro de servlet para instrumentar las solicitudes entrantes.
- Está utilizando clientes instrumentados fuera del código servlet: si utiliza un cliente instrumentado para realizar llamadas en código de inicio u otro tipo de código que no se ejecute en respuesta a una solicitud entrante, debe crear un segmento manualmente. Para ver ejemplos, consulte Instrumentación de código de inicio.
- Está usando clientes instrumentados en subprocesos de procesos de trabajo: al crear un nuevo subproceso, la grabadora de X-Ray pierde su referencia al segmento abierto. Puede utilizar los métodos `getTraceEntity` y `setTraceEntity` para obtener una referencia al segmento o subsegmento actual (`Entity`) y devolverlo a la grabadora dentro del hilo. Consulte Uso de clientes instrumentados en subprocesos de trabajo para ver un ejemplo.

SDK de X-Ray para Node.js

Problema: CLS no funciona con Sequelize

Pase el espacio de nombres del SDK de X-Ray para Node.js a Sequelize con el método `cls`.

```
var AWSXRay = require('aws-xray-sdk');
```

```
const Sequelize = require('sequelize');
Sequelize.cls = AWSXRay.getNamespace();
const sequelize = new Sequelize(...);
```

Problema: CLS no funciona con Bluebird

Utilice `cls-bluebird` para que Bluebird funcione con CLS.

```
var AWSXRay = require('aws-xray-sdk');
var Promise = require('bluebird');
var clsBluebird = require('cls-bluebird');
clsBluebird(AWSXRay.getNamespace());
```

El daemon de X-Ray

Problema: El demonio está utilizando credenciales incorrectas

El daemon utiliza el SDK de AWS para cargar las credenciales. Si se emplean varios métodos para proporcionar las credenciales, se utilizará el método que tenga la máxima prioridad. Para obtener más información, consulte Ejecutar el demonio.

Historial de documentos para AWS X-Ray

En la siguiente tabla se describen los cambios importantes en la documentación de AWS X-Ray. Para recibir notificaciones sobre los cambios en esta documentación, puede suscribirse a una fuente RSS.

Última actualización de la documentación: 7 de marzo de 2024

Cambio	Descripción	Fecha
Cronología actualizada de soporte para AWS X-Ray SDKs y Daemon	El 25 de febrero de 2026, el AWS X-Ray SDKs/Daemon entrará en modo de mantenimiento, donde AWS se limitarán las versiones de X-Ray SDK y Daemon para abordar únicamente problemas de seguridad. Para obtener más información sobre el cronograma de soporte, consulte el cronograma de soporte de X-Ray SDK y Daemon. Se recomienda migrar a. OpenTelemetry Para obtener más información sobre la migración a OpenTelemetry, consulte Migración de una instrumentación de rayos X a una instrumentación. OpenTelemetry	26 de noviembre de 2025
end-of-supportAviso agregado para AWS X-Ray SDKs y Daemon	El 25 de febrero de 2027, AWS X-Ray dejará de ofrecer soporte a AWS X-Ray SDKs y Daemon. Se recomienda migrar a. OpenTelemetry	22 de agosto de 2025

	Para obtener más informaci ón, consulte Migración de una instrumentación de rayos X a OpenTelemetry una instrumen tación.	
Funcionalidad agregada	Migración de X-Ray a. OpenTelemetry Para obtener más información, consulte Migración de una instrumentación de rayos X a OpenTelemetry una instrumen tación.	13 de junio de 2025
Funcionalidad agregada	AWS X-Ray ahora admite la búsqueda de transacciones. Para obtener más información, consulte Transaction Search.	21 de noviembre de 2024
Funcionalidad agregada	AWS X-Ray ahora es compatible con OpenTelem etry Protocol (OTLP) Endpoint. Para obtener más información, consulte OpenTelemetry.	21 de noviembre de 2024

Funcionalidad agregada	X-Ray ahora registra eventos de datos `PutTraceSegments`, incluidos `GetTraceSummaries`, y `BatchGetTraces` para AWS CloudTrail. X-Ray ahora también registra el evento `GetSamplingStatisticSummaries` de administración en CloudTrail. Para obtener más información, consulte Registrar llamadas a la API X-Ray con AWS CloudTrail.	7 de marzo de 2024
Funcionalidad agregada	X-Ray ahora admite el rastreo IDs creado mediante OpenTelemetry o cualquier otro marco que cumpla con la especificación Trace Context del W3C. Para obtener más información, consulte Envío de datos de rastro a X-Ray.	25 de octubre de 2023
Funcionalidad agregada	A partir de ahora, puede configurar el rastreo activo de Amazon SNS, lo que le permite rastrear y analizar las solicitudes a medida que avanzan a través de los temas de Amazon SNS. Para obtener más información, consulte Amazon SNS y. AWS X-Ray	8 de febrero de 2023

Tema sobre el SDK de X-Ray para Node.js actualizado	Se agregaron detalles para la instrumentación de los clientes que utilizan la V3. AWS SDK para JavaScript Para obtener más información, consulte Rastreo de llamadas del AWS SDK con el SDK de X-Ray para Node.js.	7 de febrero de 2023
Detalles de la política administrada de IAM actualizados	Se agregó el permiso de IAM para la observabilidad entre cuentas a las políticas administradas `AWSXRayReadOnlyAccess` , `AWSXRayFullAccess` y `AWSXrayCrossAccountSharingConfiguration` . Para obtener más información, consulte Políticas administradas de IAM para X-Ray.	7 de febrero de 2023
Funcionalidad agregada	AWS X-Ray ahora es compatible con la observabilidad entre cuentas, lo que le permite supervisar y solucionar problemas de las aplicaciones que abarcan varias cuentas dentro de una. Región de AWS Para obtener más información, consulte Rastreo entre cuentas.	27 de noviembre de 2022

Funcionalidad agregada	Ahora puede ver los rastros vinculados entre los productores de mensajes, una cola de Amazon SQS y los consumidores, lo que proporciona una vista conectada de los rastros enviados desde aplicaciones basadas en eventos. Para obtener más información, consulte Rastreo de aplicaciones basadas en eventos.	20 de noviembre de 2022
Detalles de la política administrada de IAM actualizados	Se agregó el permiso de IAM para incluir políticas de recursos en la política administrada `AWSXRayReadOnlyAccess`. Para obtener más información, consulte Políticas administradas de IAM para X-Ray.	15 de noviembre de 2022
Permisos de la consola de IAM y detalles de la política administrada actualizados	Se actualizó el conjunto de permisos de IAM que utiliza la consola de X-Ray, junto con la descripción de la política administrada `AWSXRayReadOnlyAccess`. Para obtener más información, consulte Uso de la consola X-Ray.	11 de noviembre de 2022

Se agregó AWS una distribución para Ruby OpenTelemetry	AWS Distro for OpenTelemetry (ADOT) proporciona un conjunto único de código abierto APIs, bibliotecas y agentes para recopilar rastreos y métricas distribuidos. ADOT Ruby le permite instrumentar su aplicación Ruby para X-Ray y otros backends de rastreo. Para obtener más información, consulte AWS Distro for Ruby. OpenTelemetry	7 de febrero de 2022
Funcionalidad agregada	Ahora puede ver las trazas y configurar X-Ray desde la CloudWatch consola. Para obtener más información, consulte Consola de X-Ray.	24 de enero de 2022
CloudWatch RUM integrado	Con AWS X-Ray el CloudWatch RUM, puede analizar y depurar la ruta de las solicitudes empezando por los usuarios finales de su aplicación y pasando por los servicios AWS gestionados posteriores. Para obtener más información, consulte CloudWatch RUM y. AWS X-Ray	3 de diciembre de 2021

AWS Distribución integrada para OpenTelemetry	The AWS Distro for OpenTelemetry (ADOT) proporciona un conjunto único de código abierto APIs, bibliotecas y agentes para recopilar rastreos y métricas distribuidos. ADOT le permite instrumentar su aplicación para X-Ray y otros backends de rastreo. Para obtener más información, consulte Instrumentación de su aplicación.	23 de septiembre de 2021
Funcionalidad agregada	AWS X-Ray ahora se integra con Amazon Virtual Private Cloud, lo que permite que los recursos de su Amazon VPC se comuniquen con el servicio de X-Ray sin tener que pasar por la Internet pública. Para obtener más información, consulte Uso AWS X-Ray con puntos de enlace de VPC.	20 de mayo de 2021
Funcionalidad agregada	AWS X-Ray ahora se integra con CloudFormation, lo que le permite aprovisionar y configurar los recursos de X-Ray. Para obtener más información, consulte Creación de recursos de X-Ray con CloudFormation.	6 de mayo de 2021

Funcionalidad agregada	AWS X-Ray ahora se integra con Amazon EventBridge para rastrear los eventos que se transmiten EventBridge. Eso proporciona a los usuarios una visión más completa de su sistema. Para obtener más información, consulte Amazon EventBridge y AWS X-Ray.	2 de marzo de 2021
Daemon agregado a ECR	Ahora el daemon se puede descargar de Amazon ECR. Para obtener más informaci ón, consulte Descarga del daemon.	1 de marzo de 2021
Funcionalidad agregada	AWS X-Ray ahora es compatible con las notificac iones relacionadas con la información a Amazon EventBridge. Esto le permite tomar medidas automátic as sobre la información que utiliza EventBridge. Para obtener más información, consulte Notificaciones de información.	15 de octubre de 2020
Daemons descargables agregados	AWS X-Ray presenta el daemon de soporte para Linux. ARM64 Para obtener más información, consulte AWS X-Ray daemonbrazil ws	1 de octubre de 2020

Funcionalidad agregada	AWS X-Ray ahora admite la integración activa con Amazon CloudWatch Synthetics. Eso le permite ver detalles sobre un nodo cliente de valores controlados de Synthetics, como el tiempo de respuesta y el estado. También puede realizar análisis en la consola de Analytics en función de la información de un nodo cliente de valores controlados de Synthetics. Para obtener más información, consulte Depuración de CloudWatch canarios sintéticos mediante X-Ray.	24 de septiembre de 2020
Funcionalidad agregada	AWS X-Ray ahora admite flujos de trabajo de rastreo end-to-end para. AWS Step Functions Puede visualizar los componentes de su máquina de estado, identificar cuellos de botella en el rendimiento y solucionar problemas de solicitudes que dieron lugar a un error. Para obtener más información, consulte AWS Step Functions y. AWS X-Ray	14 de septiembre de 2020

Funcionalidad agregada	AWS X-Ray presenta información para analizar continuamente los datos de rastreo de su cuenta a fin de identificar problemas emergentes en sus aplicaciones. La información consiste en registros de incidentes y un seguimiento del impacto de los incidentes hasta su resolución. Para obtener más información, consulte Uso de la información en la consola AWS X-Ray	3 de septiembre de 2020
Funcionalidad agregada	AWS X-Ray presenta el agente de autoinstrumentación de Java, que permite a los clientes recopilar datos de rastreo sin tener que modificar la aplicación existente basada en Java. Ahora puede rastrear las aplicaciones web creadas en Java y basadas en servlets con un cambio mínimo de la configuración y sin cambios de código. Para obtener más información, consulte Agente de instrumentación automática de AWS X-Ray para Java.	3 de septiembre de 2020

Funcionalidad agregada	AWS X-Ray ha añadido una nueva página de grupos a la consola de X-Ray para facilitar la creación y la gestión de grupos de trazas. Para obtener más información consulte Configuración de grupos en la consola de X-Ray.	24 de agosto de 2020
Funcionalidad agregada	AWS X-Ray ahora permite añadir etiquetas a los grupos y a las reglas de muestreo. También permite controlar el acceso a los grupos y a las reglas de muestreo en función de las etiquetas. Para obtener más información, consulte Etiquetado de reglas de muestreo y grupos de X-Ray y Administración del acceso a grupos y a reglas de muestreo en función de las etiquetas.	24 de agosto de 2020

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.