aws

Entwicklerhandbuch

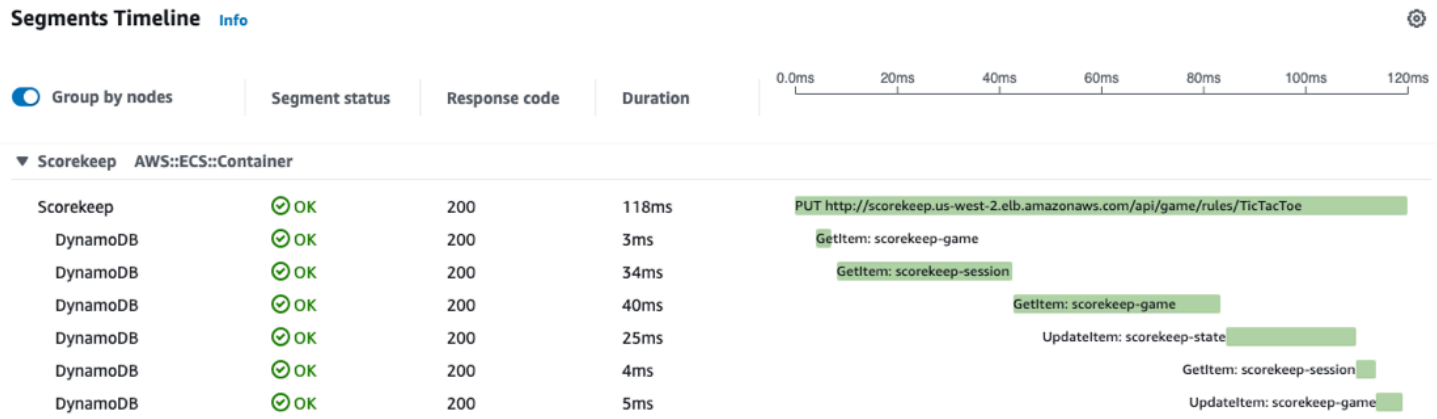
AWS X-Ray

AWS X-Ray: Entwicklerhandbuch

Table of Contents

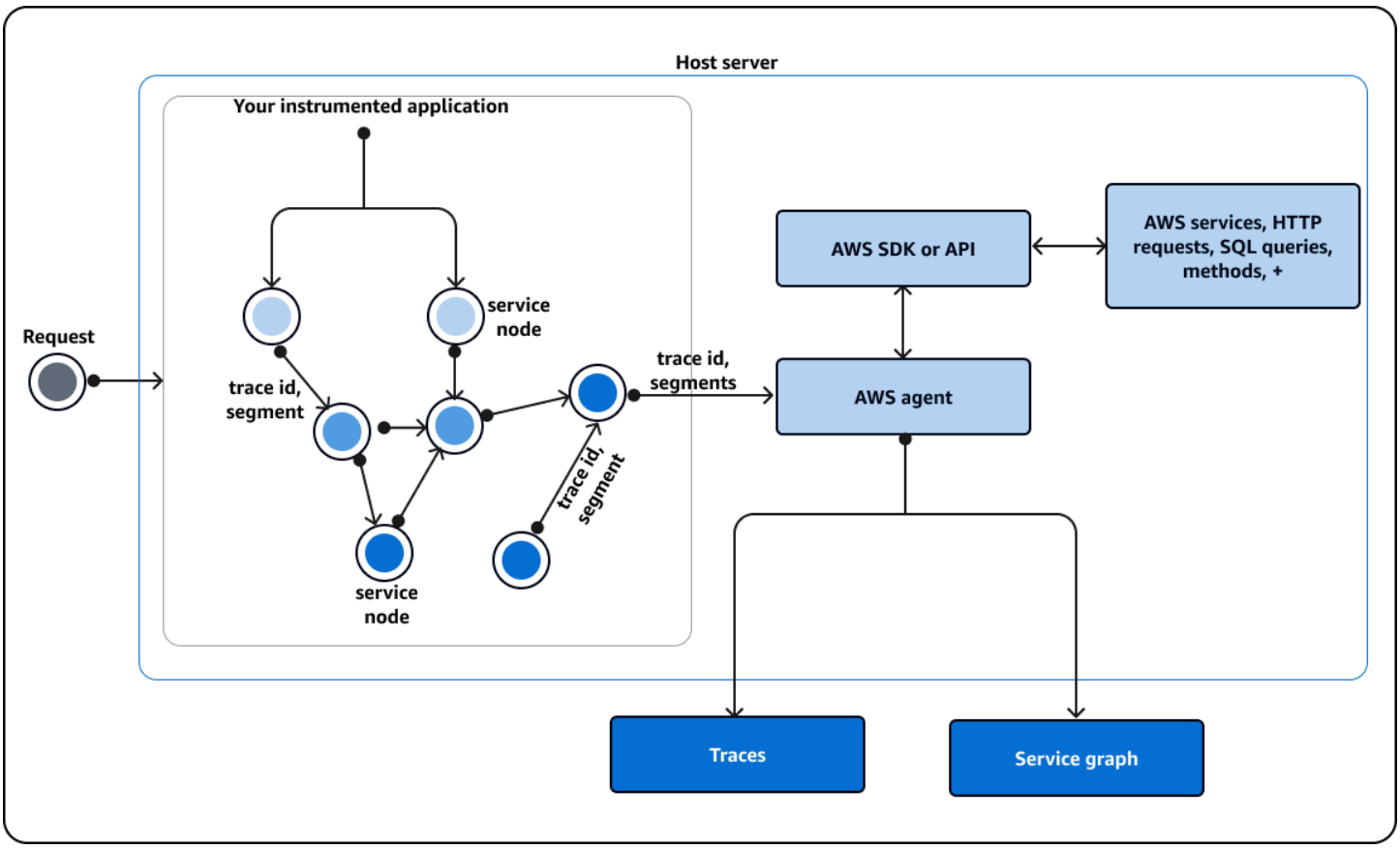
Was ist AWS X-Ray?

AWS X-Ray ist ein Dienst, der Daten über Anfragen sammelt, die Ihre Anwendung bearbeitet, und Tools bereitstellt, mit denen Sie diese Daten anzeigen, filtern und Einblicke in sie gewinnen können, um Probleme und Optimierungsmöglichkeiten zu identifizieren. Zu jeder verfolgten Anfrage an Ihre Anwendung können Sie detaillierte Informationen nicht nur über die Anfrage und Antwort abrufen, sondern auch über Aufrufe, die Ihre Anwendung an nachgelagerte AWS Ressourcen, Microservices, Datenbanken und das Internet tätigt. APIs



AWS X-Ray empfängt zusätzlich zu AWS-Services Ihren Anwendungsnutzungen, die bereits in X-Ray integriert sind, Traces von Ihrer Anwendung. Die Instrumentierung Ihrer Anwendung beinhaltet das Senden von Trace-Daten für eingehende und ausgehende Anfragen und andere Ereignisse innerhalb Ihrer Anwendung sowie Metadaten zu jeder Anfrage. Viele Instrumentierungsszenarios erfordern nur Konfigurationsänderungen. Sie können beispielsweise alle eingehenden HTTP-Anfragen und Downstream-Aufrufe Ihrer Java-Anwendung instrumentieren. AWS-Services Es gibt mehrere SDKs Agenten und Tools, mit denen Sie Ihre Anwendung für die Röntgenverfolgung instrumentieren können. Weitere Informationen finden Sie unter Instrumentierung Ihrer Anwendung.

AWS-Services die in X-Ray integriert sind, können eingehenden Anfragen Tracing-Header hinzufügen, Trace-Daten an X-Ray senden oder den X-Ray-Daemon ausführen. Sie AWS Lambda können beispielsweise Trace-Daten über Anfragen an Ihre Lambda-Funktionen senden und den X-Ray-Daemon auf Workern ausführen, um die Verwendung des X-Ray-SDK zu vereinfachen.



Anstatt Trace-Daten direkt an X-Ray zu senden, sendet jedes Client-SDK JSON-Segmentdokumente an einen Daemon-Prozess, der auf UDP-Verkehr wartet. Der X-Ray-Daemon puffert Segmente in einer Warteschlange und lädt sie stapelweise auf X-Ray hoch. Der Daemon ist für Linux, Windows und macOS verfügbar AWS Elastic Beanstalk und auf allen AWS Lambda Plattformen enthalten.

X-Ray verwendet Trace-Daten aus den AWS Ressourcen, die Ihre Cloud-Anwendungen unterstützen, um eine detaillierte Trace-Map zu erstellen. Die Trace-Map zeigt den Client, Ihren Frontend-Dienst und die Back-End-Dienste, die Ihr Frontend-Dienst aufruft, um Anfragen zu verarbeiten und Daten zu speichern. Verwenden Sie die Trace-Map, um Engpässe, Latenzspitzen und andere Probleme zu identifizieren, die Sie lösen müssen, um die Leistung Ihrer Anwendungen zu verbessern.


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

Erste Schritte mit X-Ray

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Gehen Sie wie folgt vor, um X-Ray zu verwenden:

1. Instrumentieren Sie Ihre Anwendung, sodass X-Ray verfolgen kann, wie Ihre Anwendung eine Anfrage bearbeitet.

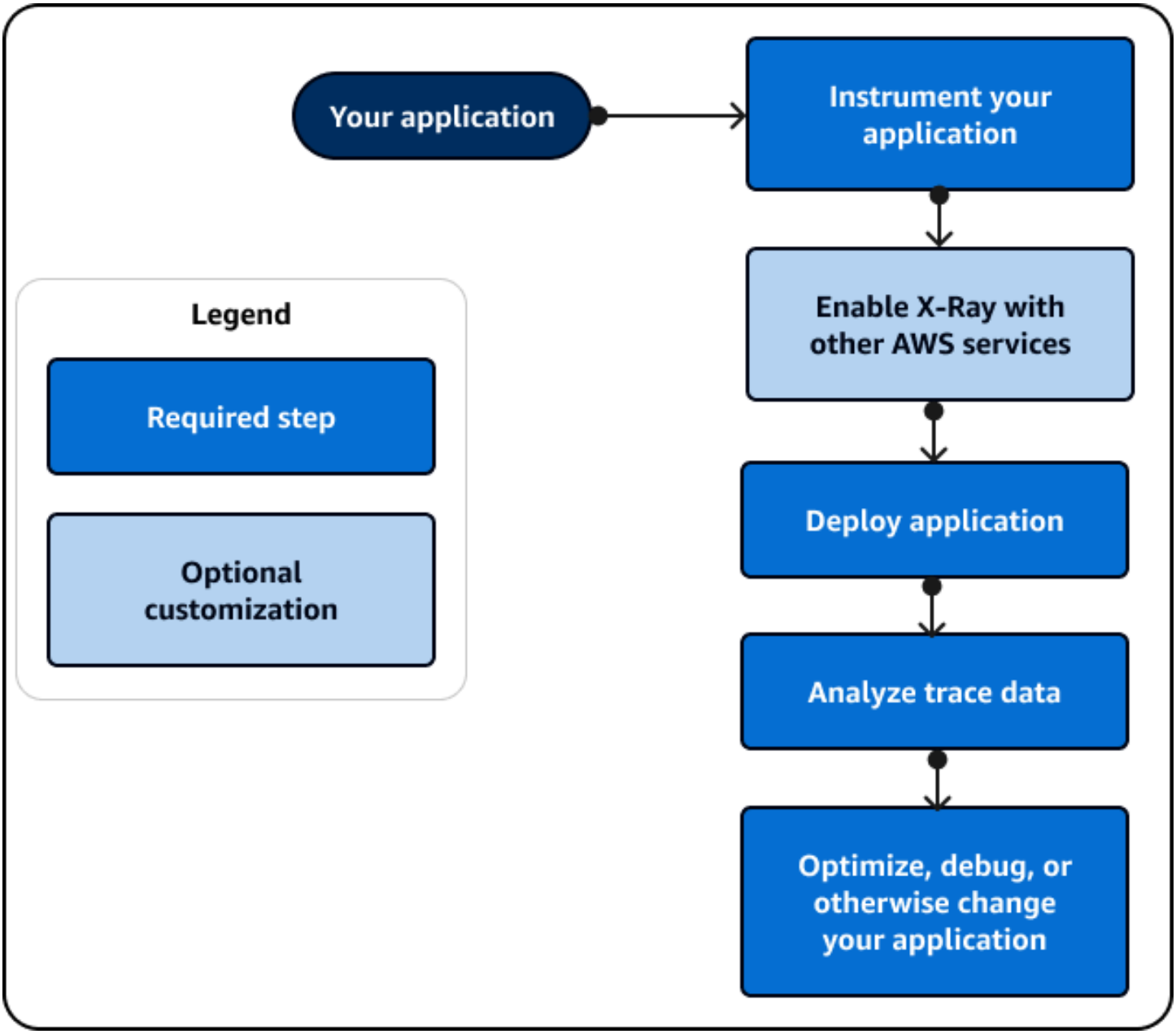
   - Verwenden Sie X-Ray ADOT - SDKs APIs, X-Ray- oder CloudWatch Anwendungssignale, um Trace-Daten an X-Ray zu senden. Weitere Informationen darüber, welche Schnittstelle verwendet werden soll, finden Sie unterAuswahl einer Schnittstelle.

   Weitere Hinweise zur Instrumentierung finden Sie unterInstrumentierung Ihrer Anwendung für AWS X-Ray.
2. (Optional) Konfigurieren Sie X-Ray so, AWS-Services dass es mit anderen Geräten funktioniert, die in X-Ray integriert sind. Sie können Traces testen und Header zu eingehenden Anfragen hinzufügen, einen Agenten oder Collector ausführen und Trace-Daten automatisch an X-Ray senden. Weitere Informationen finden Sie unter Integration AWS X-Ray mit anderen AWS-Services.
3. Stellen Sie Ihre instrumentierte Anwendung bereit. Wenn Ihre Anwendung Anfragen empfängt, zeichnet das X-Ray SDK Trace-, Segment- und Subsegmentdaten auf. In diesem Schritt müssen Sie möglicherweise auch eine IAM-Richtlinie einrichten und einen Agenten oder Collector bereitstellen.

- Beispiele für Skripts zur Bereitstellung einer Anwendung mithilfe des AWS Distro for OpenTelemetry (ADOT) -SDK und des CloudWatch Agenten auf verschiedenen Plattformen finden Sie unter Application Signals Demo Scripts.
- Ein Beispielskript für die Bereitstellung einer Anwendung mit dem X-Ray SDK und dem X-Ray-Daemon finden Sie unterAWS X-Ray Musteranwendung.

4. (Optional) Öffnen Sie eine Konsole, um die Daten anzuzeigen und zu analysieren. Sie können sich eine grafische Darstellung einer Trace-Map, einer Service Map und mehr ansehen, um zu überprüfen, wie Ihre Anwendung funktioniert. Verwenden Sie die grafischen Informationen in der Konsole, um Ihre Anwendung zu optimieren, zu debuggen und zu verstehen. Weitere Informationen zur Auswahl einer Konsole finden Sie unterBenutze eine Konsole.

Das folgende Diagramm zeigt die ersten Schritte mit X-Ray:



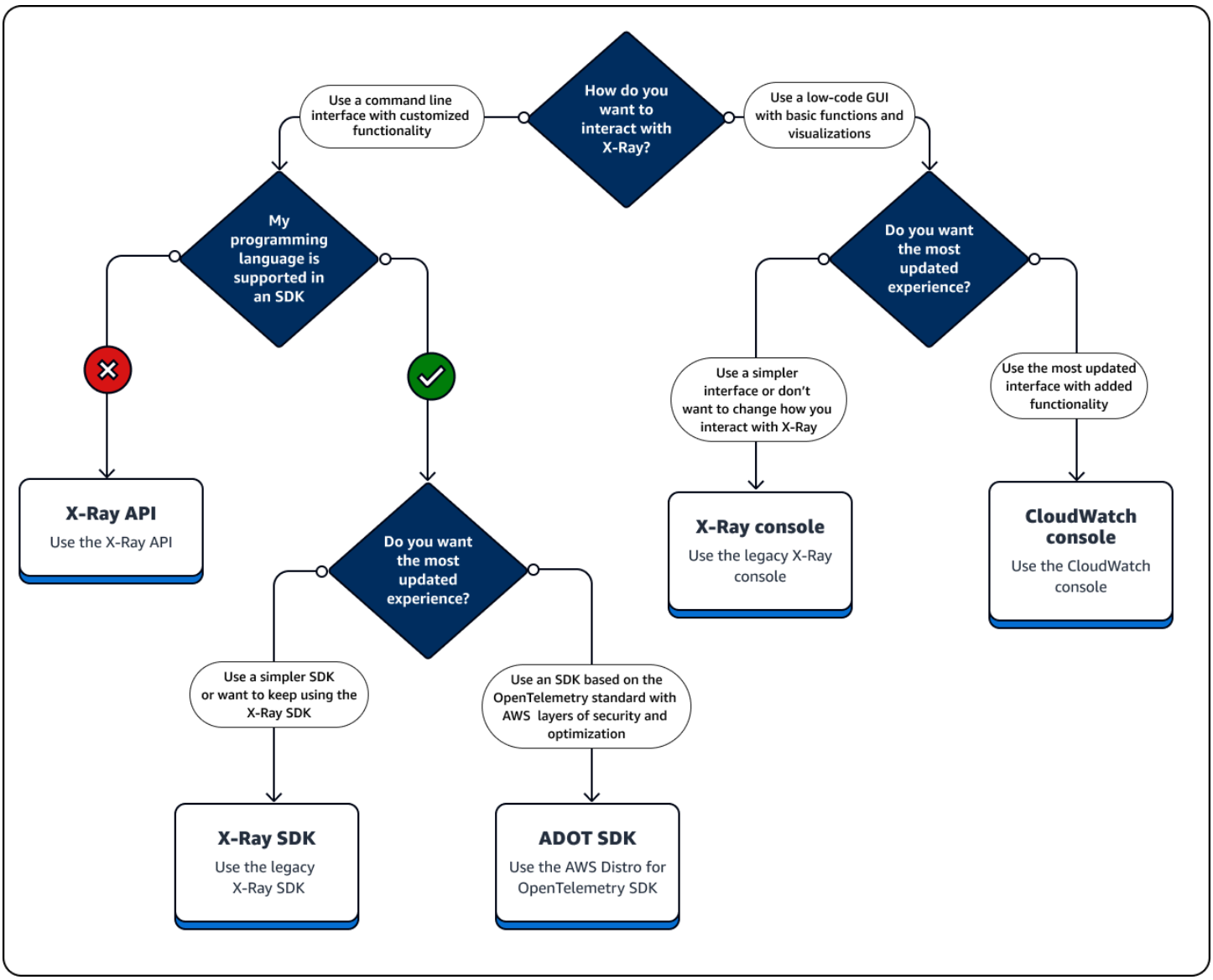
Ein Beispiel für die Daten und Karten, die in der Konsole verfügbar sind, erhalten Sie, wenn Sie eine Beispielanwendung starten, die bereits für die Generierung von Trace-Daten ausgestattet ist. In wenigen Minuten können Sie Traffic generieren, Segmente an X-Ray senden und eine Trace- und Servicekarte anzeigen.

Auswahl einer Schnittstelle

AWS X-Ray kann Ihnen Einblicke geben, wie Ihre Anwendung funktioniert und wie gut sie mit anderen Diensten und Ressourcen interagiert. Nachdem Sie Ihre Anwendung instrumentiert oder konfiguriert haben, sammelt X-Ray Trace-Daten, während Ihre Anwendung Anfragen bearbeitet. Sie können diese Trace-Daten analysieren, um Leistungsprobleme zu identifizieren, Fehler zu beheben und Ihre Ressourcen zu optimieren. Diese Anleitung zeigt Ihnen, wie Sie mit X-Ray anhand der folgenden Richtlinien interagieren:

- Verwenden Sie eine, AWS-Managementkonsole wenn Sie schnell loslegen möchten oder vorgefertigte Visualisierungen für grundlegende Aufgaben verwenden können.
  - Wählen Sie die CloudWatch Amazon-Konsole für die aktuellste Benutzererfahrung, die alle Funktionen der X-Ray-Konsole enthält.
  - Verwenden Sie die X-Ray-Konsole, wenn Sie eine einfachere Oberfläche wünschen oder die Art und Weise, wie Sie mit X-Ray interagieren, nicht ändern möchten.
- Verwenden Sie ein SDK, wenn Sie mehr benutzerdefinierte Funktionen für die Ablaufverfolgung, Überwachung oder Protokollierung benötigen, als Ihnen ein anderes bieten AWS-Managementkonsole kann.
  - Wählen Sie das ADOT SDK, wenn Sie ein herstellerunabhängiges SDK benötigen, das auf dem OpenTelemetry Open-Source-SDK basiert und zusätzliche Sicherheits- und Optimierungsebenen bietet. AWS
  - Wählen Sie das X-Ray-SDK, wenn Sie ein einfacheres SDK wünschen oder Ihren Anwendungscode nicht aktualisieren möchten.
- Verwenden Sie X-Ray-API-Operationen, wenn ein SDK die Programmiersprache Ihrer Anwendung nicht unterstützt.

Das folgende Diagramm hilft Ihnen bei der Auswahl, wie Sie mit X-Ray interagieren möchten:



Erkunden Sie die Schnittstellentypen

- Verwenden Sie ein SDK
- Benutze eine Konsole
- Verwenden Sie die X-Ray-API

Verwenden Sie ein SDK

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Verwenden Sie ein SDK, wenn Sie eine Befehlszeilenschnittstelle verwenden möchten oder mehr benutzerdefinierte Tracing-, Überwachungs- oder Protokollierungsfunktionen benötigen als die, die in einem verfügbar sind. AWS-Managementkonsole Sie können auch ein AWS SDK verwenden, um Programme zu entwickeln, die X-Ray verwenden APIs. Sie können entweder das AWS Distro for OpenTelemetry (ADOT) SDK oder das X-Ray SDK verwenden.

Wenn Sie ein SDK verwenden, können Sie Ihrem Workflow sowohl bei der Instrumentierung Ihrer Anwendung als auch bei der Konfiguration Ihres Collectors oder Agenten Anpassungen hinzufügen. Sie können ein SDK für die folgenden Aufgaben verwenden, die Sie mit einem AWS-Managementkonsole nicht erledigen können:

- Veröffentlichen Sie benutzerdefinierte Metriken — Stichproben von Metriken mit hoher Auflösung von bis zu 1 Sekunde, verwenden Sie mehrere Dimensionen, um Informationen zu einer Metrik hinzuzufügen, und aggregieren Sie Datenpunkte in einem Statistiksatz.
- Passen Sie Ihren Collector an — Passen Sie die Konfiguration für jeden Teil eines Collectors an, einschließlich Empfänger, Prozessor, Exporter und Konnektor.
- Passen Sie Ihre Instrumentierung an — Passen Sie Segmente und Untersegmente an, fügen Sie benutzerdefinierte Schlüssel-Wert-Paare als Attribute hinzu und erstellen Sie benutzerdefinierte Metriken.
- Erstellen und aktualisieren Sie Stichprobenregeln programmgesteuert.

Verwenden Sie das ADOT SDK, wenn Sie die Flexibilität eines standardisierten OpenTelemetry SDK mit zusätzlichen AWS Sicherheits- und Optimierungsebenen nutzen möchten. Das AWS

Distro for OpenTelemetry (ADOT) SDK ist ein herstellerunabhängiges Paket, das die Integration mit Backends von anderen Anbietern und AWS Nichtdiensten ermöglicht, ohne dass Sie Ihren Code neu instrumentieren müssen.

Verwenden Sie das X-Ray-SDK, wenn Sie das X-Ray-SDK bereits verwenden, nur in AWS Backends integrieren und die Art und Weise, wie Sie mit X-Ray oder Ihrem Anwendungscode interagieren, nicht ändern möchten.

Weitere Informationen zu den einzelnen Funktionen finden Sie unterWahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs.

Verwenden Sie das ADOT SDK

Das ADOT SDK besteht aus einer Reihe von Open-Source-Bibliotheken und -Agenten APIs, die Daten an Backend-Dienste senden. ADOTwird von mehreren Backends und Agenten unterstützt AWS, lässt sich in mehrere Backends und Agenten integrieren und bietet eine große Anzahl von Open-Source-Bibliotheken, die von der OpenTelemetry Community verwaltet werden. Verwenden Sie das ADOT SDK, um Ihre Anwendung zu instrumentieren und Protokolle, Metadaten, Metriken und Traces zu sammeln. Sie können es auch verwendenADOT, um Dienste zu überwachen und einen Alarm auf der Grundlage Ihrer Messwerte in einzustellen CloudWatch.

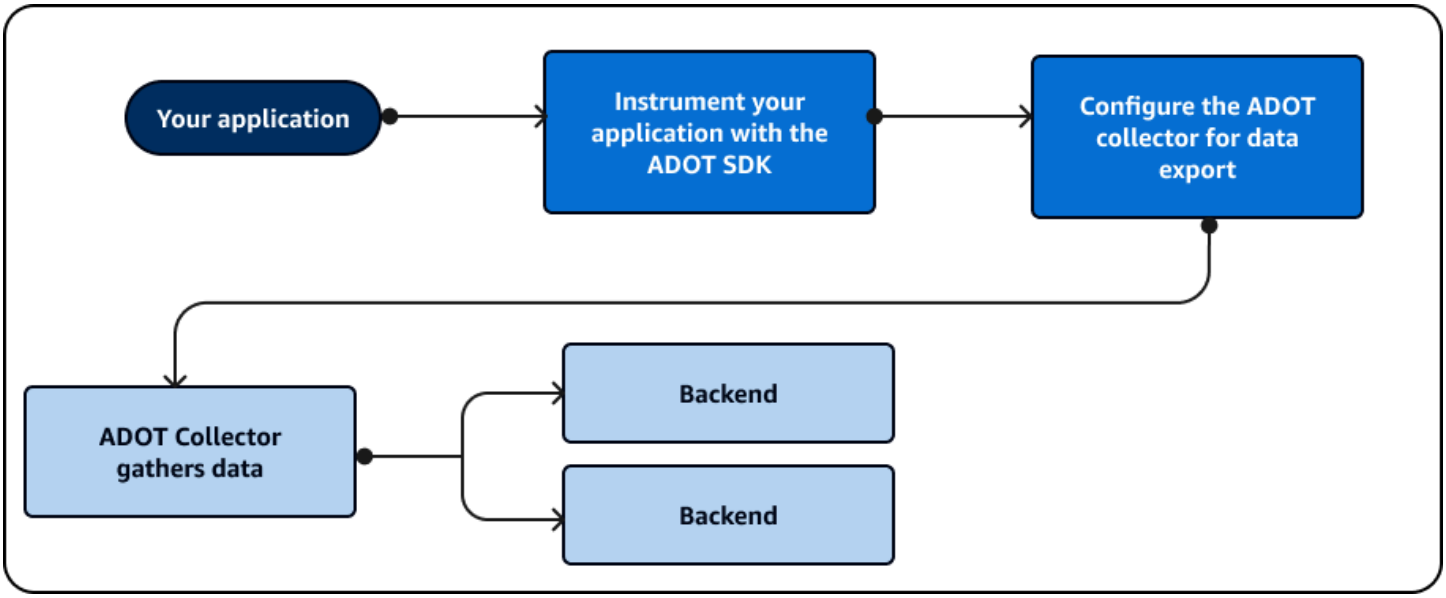
Wenn Sie das ADOT SDK verwenden, haben Sie in Kombination mit einem Agenten die folgenden Optionen:

- Verwenden Sie das ADOT SDK mit dem CloudWatch Agenten — empfohlen.
- Verwenden Sie das ADOT SDK mit dem ADOTCollector — empfohlen, wenn Sie herstellerunabhängige Software mit Sicherheits- und AWS Optimierungsebenen verwenden möchten.

Gehen Sie wie folgt vor, um das ADOT SDK zu verwenden:

- Instrumentieren Sie Ihre Anwendung mithilfe des ADOT SDK. Weitere Informationen finden Sie in der Dokumentation zu Ihrer Programmiersprache in der technischen Dokumentation von ADOT.
- Konfigurieren Sie einen ADOT Collector so, dass er ihm mitteilt, wohin die gesammelten Daten gesendet werden sollen.

Nachdem der ADOT Collector Ihre Daten empfangen hat, sendet er sie an das Backend, das Sie in der ADOT Konfiguration angeben. ADOTkann Daten an mehrere Backends senden, auch an Anbieter außerhalb von AWS, wie in der folgenden Abbildung dargestellt:



AWS wird regelmäßig aktualisiertADOT, um Funktionen hinzuzufügen und sich an das OpenTelemetryFramework anzupassen. Aktualisierungen und future Entwicklungspläne ADOT sind Teil einer Roadmap, die der Öffentlichkeit zugänglich ist. ADOTunterstützt mehrere Programmiersprachen, darunter die folgenden:

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

Wenn Sie Python verwenden, ADOT kann Ihre Anwendung automatisch instrumentieren. Informationen zu den ersten Schritten ADOT finden Sie unter Einführung und Erste Schritte mit der AWS Distribution for OpenTelemetry Collector.

Verwenden Sie das SDK X-Ray

Das X-Ray-SDK besteht aus einer Reihe von AWS APIs UND-Bibliotheken, die Daten an AWS Backend-Dienste senden. Verwenden Sie das X-Ray SDK, um Ihre Anwendung zu instrumentieren und Trace-Daten zu sammeln. Sie können das X-Ray SDK nicht zum Sammeln von Protokoll- oder Metrikdaten verwenden.

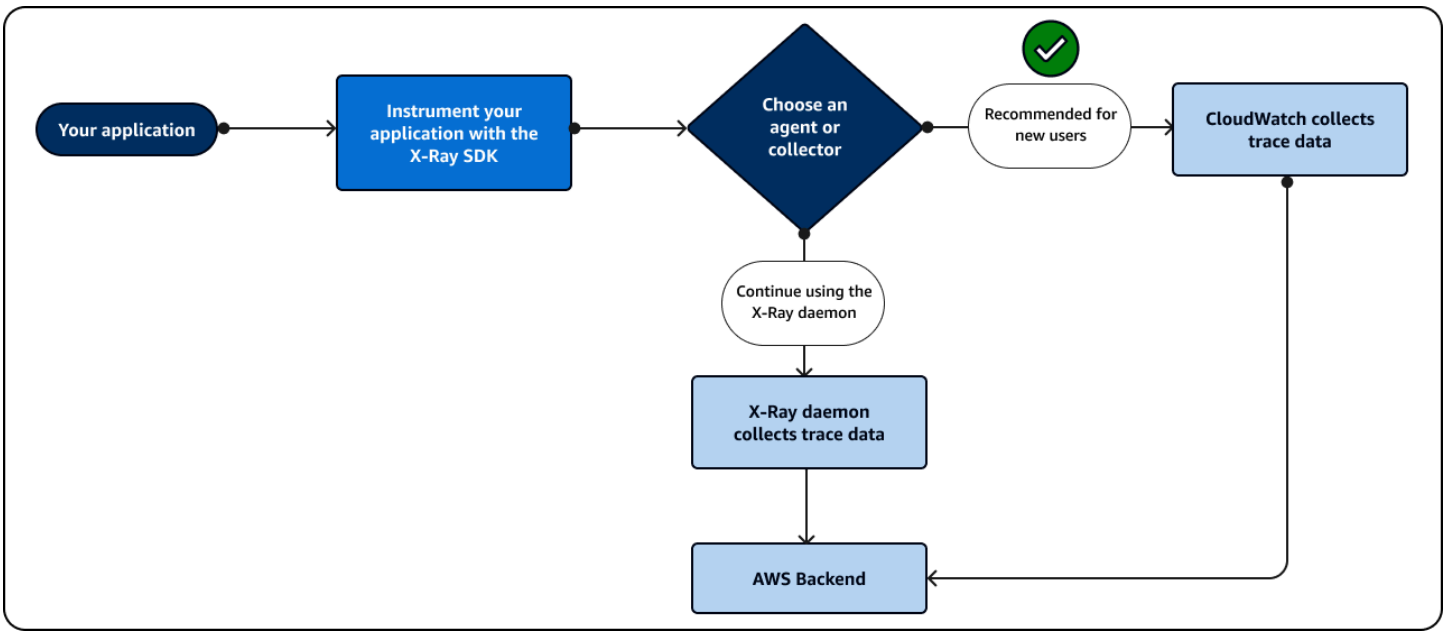
Wenn Sie das X-Ray SDK verwenden, haben Sie in Kombination mit einem Agenten die folgenden Optionen:

- Verwenden Sie das X-Ray-SDK mit AWS X-Ray Dämon — Verwenden Sie dies, wenn Sie Ihren Anwendungscode nicht aktualisieren möchten.
- Verwenden Sie das X-Ray-SDK mit dem CloudWatch Agenten — (empfohlen) Der CloudWatch Agent ist mit dem X-Ray-SDK kompatibel.

Gehen Sie wie folgt vor, um das X-Ray SDK zu verwenden:

- Instrumentieren Sie Ihre Anwendung mit dem X-Ray SDK.
- Konfigurieren Sie einen Collector so, dass er ihm mitteilt, wohin er die gesammelten Daten senden soll. Sie können entweder den CloudWatch Agenten oder den X-Ray-Daemon verwenden, um Ihre Trace-Informationen zu sammeln.

Nachdem der Collector oder Agent Ihre Daten empfangen hat, sendet er sie an ein AWS Backend, das Sie in der Agentenkonfiguration angeben. Das X-Ray SDK kann nur Daten an ein AWS Backend senden, wie in der folgenden Abbildung dargestellt:



Wenn Sie verwendenJava, können Sie das X-Ray SDK verwenden, um Ihre Anwendung automatisch zu instrumentieren. Informationen zu den ersten Schritten mit dem X-Ray-SDK finden Sie in den Bibliotheken, die den folgenden Programmiersprachen zugeordnet sind:

- Go
- Java
- Node.js
- Python
- .NET
- Ruby

Benutze eine Konsole

Verwenden Sie eine Konsole, wenn Sie eine grafische Benutzeroberfläche (GUI) benötigen, die nur wenig Codierung erfordert. Benutzer, die mit X-Ray noch nicht vertraut sind, können mithilfe vorgefertigter Visualisierungen schnell loslegen und grundlegende Aufgaben ausführen. Sie können die folgenden Aktionen direkt von der Konsole aus ausführen:

- X-Ray aktivieren.
- Sehen Sie sich allgemeine Zusammenfassungen der Leistung Ihrer Anwendung an.
- Überprüfen Sie den Status Ihrer Anwendungen.

- Identifizieren Sie Fehler auf hoher Ebene.
- Sehen Sie sich grundlegende Trace-Zusammenfassungen an.

Sie können entweder die CloudWatch Amazon-Konsole unter https://console.aws.amazon.com/cloudwatch/oder die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause verwenden, um mit X-Ray zu interagieren.

Verwenden Sie die CloudWatch Amazon-Konsole

Die CloudWatch Konsole enthält neue X-Ray-Funktionen, die gegenüber der X-Ray-Konsole neu gestaltet wurden, um die Bedienung zu vereinfachen. Wenn Sie die CloudWatch Konsole verwenden, können Sie CloudWatch Protokolle und Metriken zusammen mit X-Ray-Trace-Daten anzeigen. Verwenden Sie die CloudWatch Konsole, um Daten wie die folgenden anzuzeigen und zu analysieren:

- X-Ray-Traces — Sie können die mit Ihrer Anwendung verknüpften Traces anzeigen, analysieren und filtern, während sie eine Anfrage bearbeitet. Verwenden Sie diese Traces, um hohe Latenzen zu finden, Fehler zu debuggen und Ihren Anwendungsworkflow zu optimieren. Sehen Sie sich eine Trace Map und eine Service Map an, um visuelle Darstellungen Ihres Anwendungsworkflows zu sehen.
- Protokolle — Sie können die von Ihrer Anwendung erstellten Protokolle anzeigen, analysieren und filtern. Verwenden Sie Protokolle, um Fehler zu beheben und die Überwachung auf der Grundlage bestimmter Protokollwerte einzurichten.
- Metriken — Messen und überwachen Sie die Leistung Ihrer Anwendung anhand von Metriken, die Ihre Ressourcen ausgeben, oder erstellen Sie Ihre eigenen Messwerte. Sehen Sie sich diese Metriken in Grafiken und Diagrammen an.
- Überwachung von Netzwerken und Infrastruktur — Überwachen Sie wichtige Netzwerke auf Ausfälle und den Zustand und die Leistung Ihrer Infrastruktur, einschließlich containerisierter Anwendungen, anderer AWS Dienste und Clients.
- Alle Funktionen der X-Ray-Konsole, die im folgenden Abschnitt „Verwenden Sie die X-Ray-Konsole“ aufgeführt sind.

Weitere Informationen zur CloudWatch Konsole finden Sie unter Erste Schritte mit Amazon CloudWatch.

Loggen Sie sich in die CloudWatch Amazon-Konsole ein unter https://console.aws.amazon.com/cloudwatch/.

Verwenden Sie die X-Ray-Konsole

Die X-Ray-Konsole bietet verteiltes Tracing für Anwendungsanfragen. Verwenden Sie die X-Ray-Konsole, wenn Sie eine einfachere Konsolenerfahrung wünschen oder Ihren Anwendungscode nicht aktualisieren möchten. AWS entwickelt die X-Ray-Konsole nicht mehr. Die X-Ray-Konsole enthält die folgenden Funktionen für instrumentierte Anwendungen:

- Einblicke — Erkennen Sie automatisch Anomalien in der Leistung Ihrer Anwendung und finden Sie die zugrunde liegenden Ursachen. Insights sind in der CloudWatch Konsole unter Insights enthalten. Weitere Informationen finden Sie unter Verwenden von X-Ray Insights inVerwenden Sie die X-Ray-Konsole.
- Dienstübersicht — Zeigt eine grafische Struktur Ihrer Anwendung und ihrer Verbindungen zu Clients, Ressourcen, Diensten und Abhängigkeiten an.
- Ablaufverfolgungen — Sehen Sie sich eine Übersicht der Traces an, die von Ihrer Anwendung bei der Bearbeitung einer Anfrage generiert werden. Verwenden Sie Trace-Daten, um zu verstehen, wie Ihre Anwendung im Vergleich zu grundlegenden Kennzahlen wie HTTP Antwort- und Antwortzeit abschneidet.
- Analytik — Interpretieren, untersuchen und analysieren Sie Trace-Daten mithilfe von Grafiken zur Verteilung der Antwortzeiten.
- Konfiguration — Erstellen Sie benutzerdefinierte Traces, um die Standardkonfigurationen für Folgendes zu ändern:
  - Probenahme — Erstellen Sie eine Regel, die festlegt, wie oft Ihre Anwendung nach Trace-Informationen durchsucht werden soll. Weitere Informationen finden Sie unter Probenahmeregeln konfigurieren unterVerwenden Sie die X-Ray-Konsole.
  - Verschlüsselung — Verschlüsseln Sie Daten im Ruhezustand mit einem Schlüssel, den Sie überprüfen oder deaktivieren können. AWS Key Management Service
  - Gruppen — Verwenden Sie einen Filterausdruck, um eine Gruppe von Traces mit einem gemeinsamen Merkmal wie dem Namen einer URL oder einer Antwortzeit zu definieren. Weitere Informationen finden Sie unter Gruppen konfigurieren.

Loggen Sie sich zu https://console.aws.amazon.com/xray/Hause in die X-Ray-Konsole ein.

Erkunden Sie die X-Ray-Konsole

Verwenden Sie die X-Ray-Konsole, um eine Übersicht der Dienste und der zugehörigen Traces für Anfragen anzuzeigen, die Ihre Anwendungen bearbeiten, und um Gruppen und Sampling-Regeln zu konfigurieren, die beeinflussen, wie Traces an X-Ray gesendet werden.

> **Note**
>
> Die X-Ray-Service-Karte und die CloudWatch ServiceLens Karte wurden in der CloudWatch Amazon-Konsole zur X-Ray-Trace-Map zusammengefasst. Öffnen Sie die CloudWatchKonsole und wählen Sie im linken Navigationsbereich unter X-Ray-Traces die Option Trace Map aus.
> CloudWatch enthält jetzt Application Signals, mit denen Sie Ihre Anwendungsdienste, Clients, Synthetics-Kanarien und Serviceabhängigkeiten erkennen und überwachen können. Verwenden Sie Application Signals, um eine Liste oder eine visuelle Übersicht Ihrer Services zu erhalten, Integritätskennzahlen auf der Grundlage Ihrer Service-Level-Ziele (SLOs) einzusehen und eine Aufschlüsselung durchzuführen, um korrelierte Röntgenspuren für eine detailliertere Fehlerbehebung zu sehen.

Die primäre X-Ray-Konsolenseite ist die Trace-Map, eine visuelle Darstellung des JSON-Dienstdiagramms, das X-Ray aus den von Ihren Anwendungen generierten Trace-Daten generiert. Die Übersicht besteht aus Service-Knoten für jede Anwendung in Ihrem Konto, das Anforderungen verarbeitet, aus vorgelagerten Knoten, die die Ursprünge der Anforderungen repräsentieren, und aus nachgelagerten Service-Knoten, die Web-Services und Ressourcen darstellen, die von einer Anwendung während der Verarbeitung einer Anforderung verwendet werden. Es gibt zusätzliche Seiten zum Anzeigen von Traces und Trace-Details sowie zum Konfigurieren von Gruppen und Stichprobenregeln.

Sehen Sie sich das Konsolenerlebnis für X-Ray an und vergleichen Sie es mit der CloudWatch Konsole in den folgenden Abschnitten.

Erkunden Sie X-Ray und CloudWatch Konsolen

- Verwenden der X-Ray-Trace-Map
- Traces und Trace-Details anzeigen
- Verwenden von Filterausdrücken
- Kontenübergreifende Rückverfolgung

- Nachverfolgung ereignisgesteuerter Anwendungen
- Verwendung von Latenzhistogrammen
- Nutzung von X-Ray Insights
- Interaktion mit der -Analysekonsole
- Gruppen konfigurieren
- Konfigurieren von -Samplingregeln
- Konfiguration der adaptiven Probenahme
- Deep Linking für Konsolen

Verwenden der X-Ray-Trace-Map

Sehen Sie sich die X-Ray-Trace-Map an, um Dienste zu identifizieren, bei denen Fehler auftreten, Verbindungen mit hoher Latenz oder Traces für Anfragen, die nicht erfolgreich waren.

> **Note**
>
> CloudWatch enthält jetzt Application Signals, mit denen Sie Ihre Anwendungsservices, Clients, synthetischen Datenbanken und Dienstabhängigkeiten erkennen und überwachen können. Verwenden Sie Application Signals, um eine Liste oder eine visuelle Übersicht Ihrer Services zu erhalten, Integritätskennzahlen auf der Grundlage Ihrer Service-Level-Ziele (SLOs) einzusehen und eine Aufschlüsselung durchzuführen, um korrelierte Röntgenspuren für eine detailliertere Fehlerbehebung zu sehen.
> Die X-Ray-Servicekarte und die CloudWatch ServiceLens Karte werden in der CloudWatch Amazon-Konsole zur X-Ray-Trace-Map kombiniert. Öffnen Sie die CloudWatch Konsole und wählen Sie im linken Navigationsbereich unter X-Ray-Traces die Option Trace Map aus.

Anzeigen der Ablaufverfolgungskarte

Die Trace-Map ist eine visuelle Darstellung der Trace-Daten, die von Ihren Anwendungen generiert werden. Die Karte zeigt Dienstknoten, die Anfragen bearbeiten, vorgelagerte Clientknoten, die den Ursprung der Anfragen darstellen, und Downstream-Serviceknoten, die Webdienste und Ressourcen darstellen, die von einer Anwendung bei der Bearbeitung einer Anfrage verwendet werden.

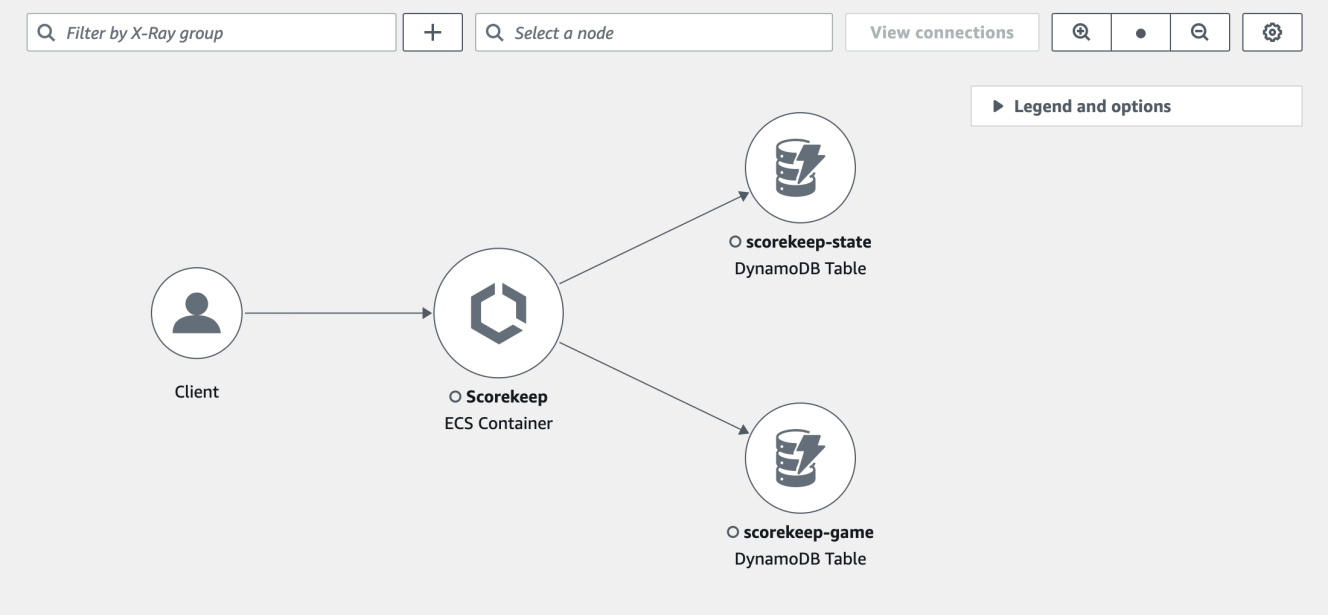
Die Trace-Map zeigt eine verknüpfte Ansicht von Traces für ereignisgesteuerte Anwendungen, die Amazon SQS und Lambda verwenden. Weitere Informationen finden Sie unter Ablaufverfolgung

ereignisgesteuerter Anwendungen. Die Trace-Map unterstützt auch die kontenübergreifende Ablaufverfolgung, sodass Knoten aus mehreren Konten in einer einzigen Map angezeigt werden.

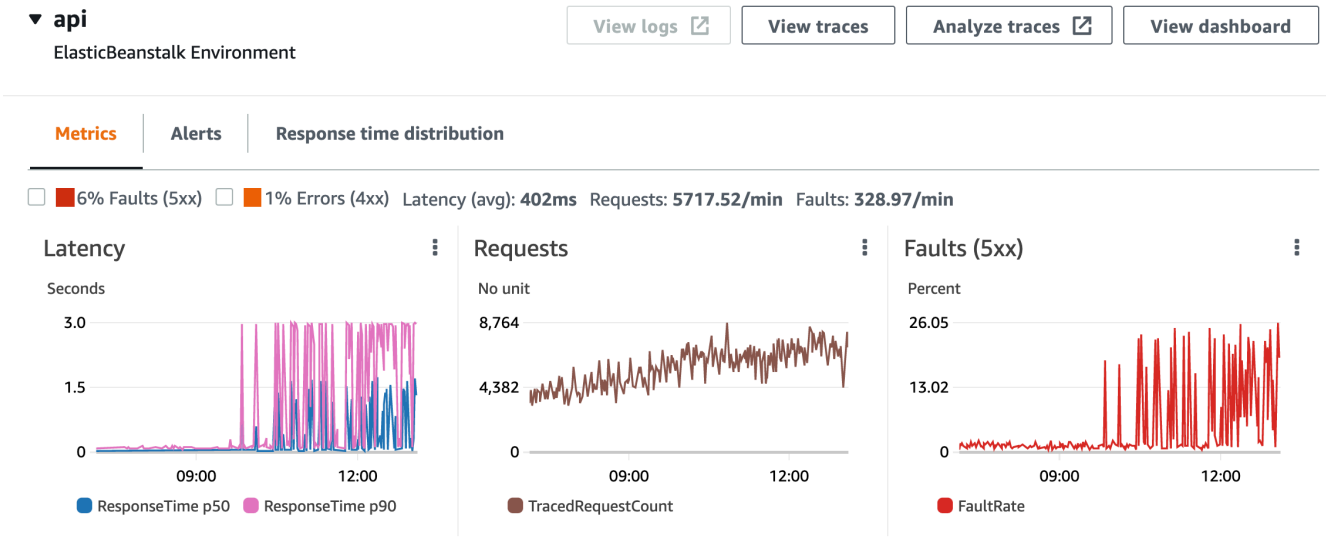
CloudWatch console

Um die Trace-Map in der Konsole anzuzeigen CloudWatch

1. Öffnen Sie die CloudWatch -Konsole. Wählen Sie im linken Navigationsbereich im Bereich X-Ray Traces die Option Trace Map aus.



2. Wählen Sie einen Service-Knoten, um Anforderungen für diesen Knoten anzuzeigen, oder eine Edge zwischen zwei Knoten, um Anforderungen anzuzeigen, die diese Verbindung passiert haben.
3. Unter der Trace-Map werden zusätzliche Informationen angezeigt, darunter Registerkarten für Messwerte, Warnmeldungen und die Verteilung der Antwortzeiten. Wählen Sie auf der Registerkarte Metriken einen Bereich innerhalb jedes Diagramms aus, um weitere Details anzuzeigen, oder wählen Sie die Optionen Fehler oder Fehler, um die Ablaufverfolgung zu filtern. Wählen Sie auf der Registerkarte Verteilung der Antwortzeiten einen Bereich innerhalb des Diagramms aus, um die Ablaufverfolgung nach Antwortzeit zu filtern.



4. Zeigen Sie Traces an, indem Sie Traces anzeigen wählen, oder wenn ein Filter angewendet wurde, wählen Sie Gefilterte Traces anzeigen.
5. Wählen Sie Protokolle anzeigen, um die mit dem ausgewählten Knoten verknüpften CloudWatch Protokolle anzuzeigen. Nicht alle Trace-Map-Knoten unterstützen das Anzeigen von Protokollen. Weitere Informationen finden Sie in den CloudWatch Protokollen zur Fehlerbehebung.

In der Trace-Map werden Probleme innerhalb der einzelnen Knoten farblich dargestellt:

- Rot für Server-Fehler (500er-Fehler)
- Gelb für Client-Fehler (400er-Fehler)
- Violett für Ablehnungsfehler („Fehler 429 – Zu viele Anfragen”)

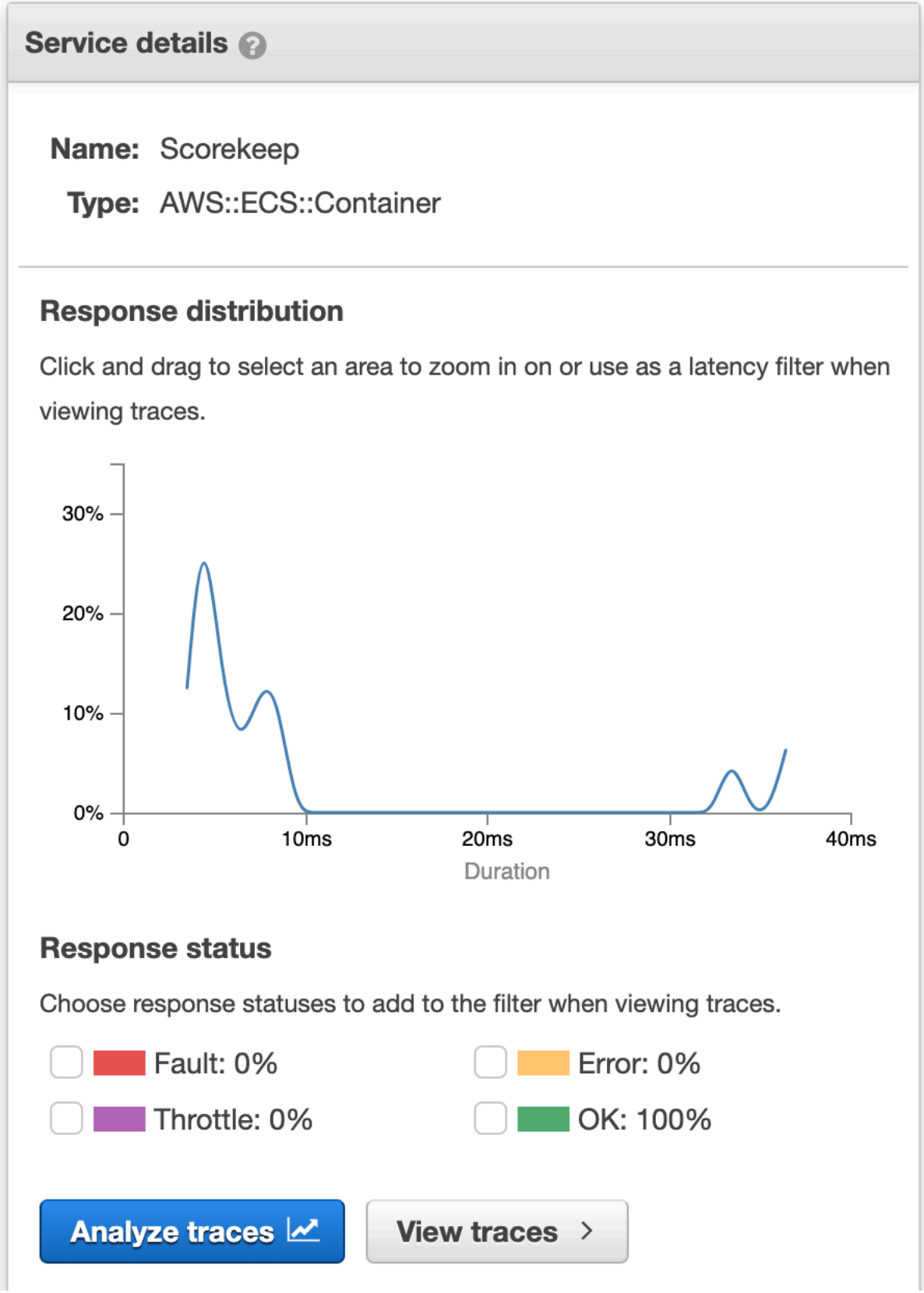
Wenn Ihre Trace-Map groß ist, verwenden Sie die Steuerelemente auf dem Bildschirm oder die Maus, um sie zu vergrößern und zu verkleinern und die Karte zu verschieben.

X-Ray console

Um die Serviceübersicht anzuzeigen

1. Öffnen Sie die X-Ray-Konsole. Die Serviceübersicht wird standardmäßig angezeigt. Sie können Service Map auch im linken Navigationsbereich auswählen.

2. Wählen Sie einen Service-Knoten, um Anforderungen für diesen Knoten anzuzeigen, oder eine Edge zwischen zwei Knoten, um Anforderungen anzuzeigen, die diese Verbindung passiert haben.
3. Verwenden Sie das Histogramm der Antwortverteilung, um Traces nach Dauer zu filtern, und wählen Sie die Statuscodes aus, für die Sie Traces anzeigen möchten. Klicken Sie anschließend auf View traces, um die Ablaufverfolgungsliste mit angewendetem Filterausdruck zu öffnen.


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

Die Service-Karte zeigt den Zustand jedes Knotens an und markiert ihn farblich auf Grundlage des Verhältnisses zwischen erfolgreichen Anrufen und Fehlern:

- Grün für erfolgreiche Anrufe
- Rot für Server-Fehler (500er-Fehler)
- Gelb für Client-Fehler (400er-Fehler)
- Violett für Ablehnungsfehler („Fehler 429 – Zu viele Anfragen”)

Wenn Ihre Service-Map groß ist, können Sie die Karte mithilfe der Steuerelemente auf dem Bildschirm oder mit der Maus vergrößern und verkleinern und verschieben.

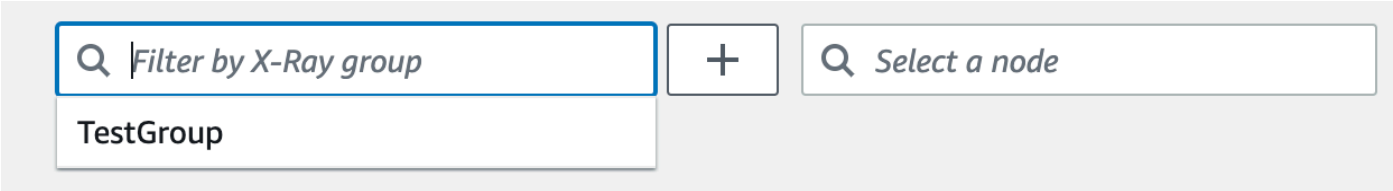
> **Note**
>
> Die X-Ray-Trace-Map kann bis zu 10.000 Knoten anzeigen. In seltenen Fällen, in denen die Gesamtzahl der Serviceknoten diesen Grenzwert überschreitet, kann es vorkommen, dass Sie eine Fehlermeldung erhalten und keine vollständige Trace-Map in der Konsole anzeigen können.

Die Trace-Map nach Gruppen filtern

Mithilfe eines Filterausdrucks können Sie Kriterien definieren, anhand derer Traces in eine Gruppe aufgenommen werden sollen. Gehen Sie wie folgt vor, um diese spezifische Gruppe dann in der Trace-Map anzuzeigen.

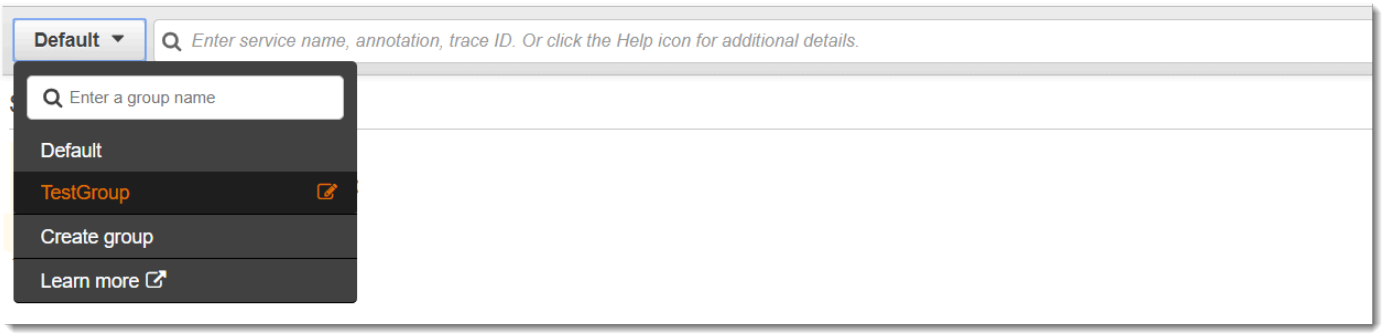
CloudWatch console

Wählen Sie einen Gruppennamen aus dem Gruppenfilter oben links auf der Trace-Map aus.



X-Ray console

Wählen Sie einen Gruppennamen aus dem Dropdown-Menü links neben der Suchleiste aus.



Die Service-Map wird nun so gefiltert, dass Traces angezeigt werden, die dem Filterausdruck der ausgewählten Gruppe entsprechen.

Legende und Optionen der Trace-Map

Die Trace-Map enthält eine Legende und mehrere Optionen zum Anpassen der Kartenanzeige.

CloudWatch console

Wählen Sie oben rechts auf der Karte das Drop-down-Menü Legende und Optionen aus. Wählen Sie aus, was in den Knoten angezeigt wird, darunter:

- Metrics zeigt die durchschnittliche Antwortzeit und die Anzahl der pro Minute während des ausgewählten Zeitraums gesendeten Traces an.
- Nodes zeigt das Dienstsymbol in jedem Knoten an.

Wählen Sie zusätzliche Karteneinstellungen aus dem Bereich Einstellungen aus, auf den Sie über das Zahnradsymbol oben rechts in der Karte zugreifen können. Zu diesen Einstellungen gehört die Auswahl, anhand welcher Metrik die Größe der einzelnen Knoten bestimmt wird und welche Kanarienvögel auf der Karte angezeigt werden sollen.

X-Ray console

Zeigen Sie die Legende der Servicekarte an, indem Sie oben rechts auf der Karte auf den Link Kartenlegende klicken. Unten rechts auf der Trace-Map können Optionen für die Servicekarte ausgewählt werden, darunter:

- Mit den Dienstsymbolen wird umgeschaltet, was innerhalb der einzelnen Knoten angezeigt wird. Dabei wird entweder das Servicesymbol oder die durchschnittliche Antwortzeit und die Anzahl der pro Minute während des ausgewählten Zeitraums gesendeten Traces angezeigt.

- Knotengröße: Bei „Keine“ werden alle Knoten auf dieselbe Größe gesetzt.
- Knotengröße: Health bewertet Knoten anhand der Anzahl der betroffenen Anfragen, einschließlich Fehlern, Störungen oder gedrosselten Anfragen.
- Knotengröße: Der Traffic berechnet die Größe der Knoten anhand der Gesamtzahl der Anfragen.

Traces und Trace-Details anzeigen

Verwenden Sie die Seite „Traces“ in der X-Ray-Konsole, um Traces anhand von URL, Antwortcode oder anderen Daten aus der Trace-Zusammenfassung zu suchen. Nachdem Sie einen Trace aus der Trace-Liste ausgewählt haben, zeigt die Seite mit den Trace-Details eine Übersicht der Service-Knoten, die dem ausgewählten Trace zugeordnet sind, sowie eine Zeitleiste mit Trace-Segmenten.

Anzeigen von Ablaufverfolgungen

CloudWatch console

Um Traces in der CloudWatch Konsole anzuzeigen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich X-Ray Traces und dann Traces aus. Sie können nach Gruppen filtern oder einen Filterausdruck eingeben. Dadurch werden die Spuren gefiltert, die im Abschnitt „Spuren“ unten auf der Seite angezeigt werden.

   Alternativ können Sie die Service-Map verwenden, um zu einem bestimmten Serviceknoten zu navigieren und sich dann die Traces anzusehen. Dadurch wird die Seite „Traces“ geöffnet, auf der bereits eine Abfrage angewendet wurde.
3. Verfeinern Sie Ihre Abfrage im Bereich Abfrageverfeinerungen. Um Traces nach einem gemeinsamen Attribut zu filtern, wählen Sie eine Option aus dem Abwärtspfeil neben Abfrage verfeinern nach. Es gibt die folgenden Optionen:

   - Knoten — Filtert Traces nach Dienstknoten.
   - Ressourcen-ARN — Filtert Traces nach einer Ressource, die einer Ablaufverfolgung zugeordnet ist. Beispiele für diese Ressourcen sind eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance, eine AWS Lambda Funktion oder eine Amazon DynamoDB Tabelle.

- Benutzer — Filtert Traces mit einer Benutzer-ID.
- Meldung zur Fehlerursache — Filtert die Ablaufverfolgung nach der Fehlerursache.
- URL — Filtert Traces nach einem URL-Pfad, der von Ihrer Anwendung verwendet wird.
- HTTP-Statuscode — Filtert Traces nach dem von Ihrer Anwendung zurückgegebenen HTTP-Statuscode. Sie können einen benutzerdefinierten Antwortcode angeben oder aus den folgenden Optionen wählen:
  - `200`— Die Anfrage war erfolgreich.
  - `401`— Der Anfrage fehlten gültige Authentifizierungsdaten.
  - `403`— Der Anfrage fehlten gültige Berechtigungen.
  - `404`— Der Server konnte die angeforderte Ressource nicht finden.
  - `500`— Der Server ist auf einen unerwarteten Fehler gestoßen und hat einen internen Fehler generiert.

Wählen Sie einen oder mehrere Einträge aus und klicken Sie dann auf Zur Abfrage hinzufügen, um den Filterausdruck oben auf der Seite zu ergänzen.

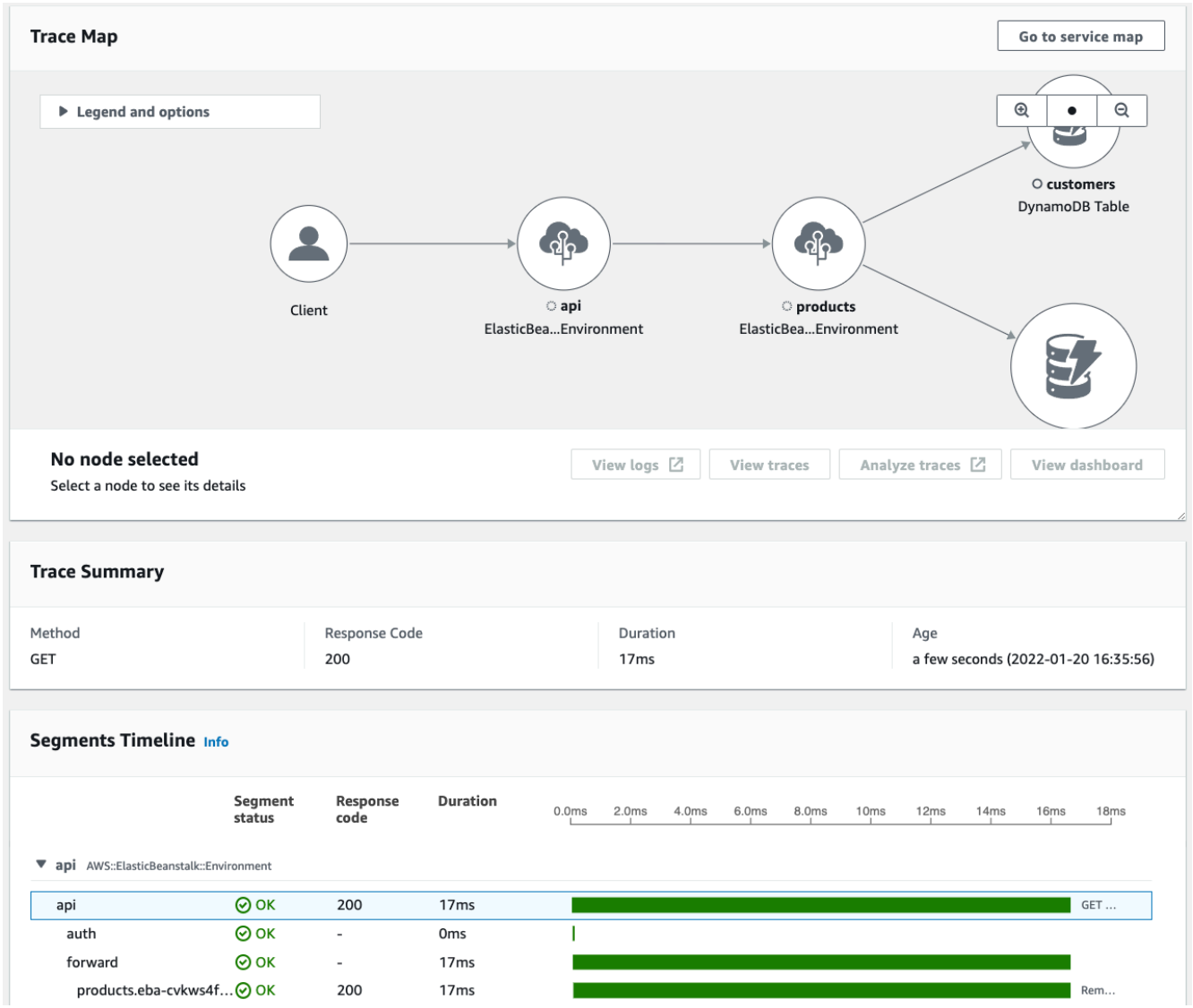
4. Um einen einzelnen Trace zu finden, geben Sie eine Trace-ID direkt in das Abfragefeld ein. Sie können das X-Ray-Format oder das World Wide Web Consortium (W3C) -Format verwenden. Ein Trace, der mit dem AWS Distro for erstellt wurde, hat beispielsweise das OpenTelemetry W3C-Format.

   > **Note**
   >
   > Wenn Sie Traces abfragen, die mit einer Trace-ID im W3C-Format erstellt wurden, zeigt die Konsole die entsprechende Ablaufverfolgung im X-Ray-Format an. Wenn Sie beispielsweise `4efaaf4d1e8720b39541901950019ee5` im W3C-Format abfragen, zeigt die Konsole das X-Ray-Äquivalent an:`1-4efaaf4d-1e8720b39541901950019ee5`.

5. Wählen Sie „Abfrage jederzeit ausführen“, um im Bereich „Traces“ unten auf der Seite eine Liste der passenden Traces anzuzeigen.
6. Um die Seite mit den Trace-Details für einen einzelnen Trace anzuzeigen, wählen Sie eine Trace-ID aus der Liste aus.

   Die folgende Abbildung zeigt eine Trace-Map mit Serviceknoten, die mit der Trace verknüpft sind, und Kanten zwischen den Knoten, die den Pfad der Segmente, aus denen sich die

Trace zusammensetzt, darstellen. Auf die Trace-Map folgt eine Trace-Zusammenfassung. Die Zusammenfassung enthält Informationen über einen GET Beispielvorgang, seinen Antwortcode, die Dauer der Ausführung der Ablaufverfolgung und das Alter der Anfrage. Die Segment-Timeline folgt der Trace-Zusammenfassung, in der die Dauer der Trace-Segmente und -Untersegmente angezeigt wird.



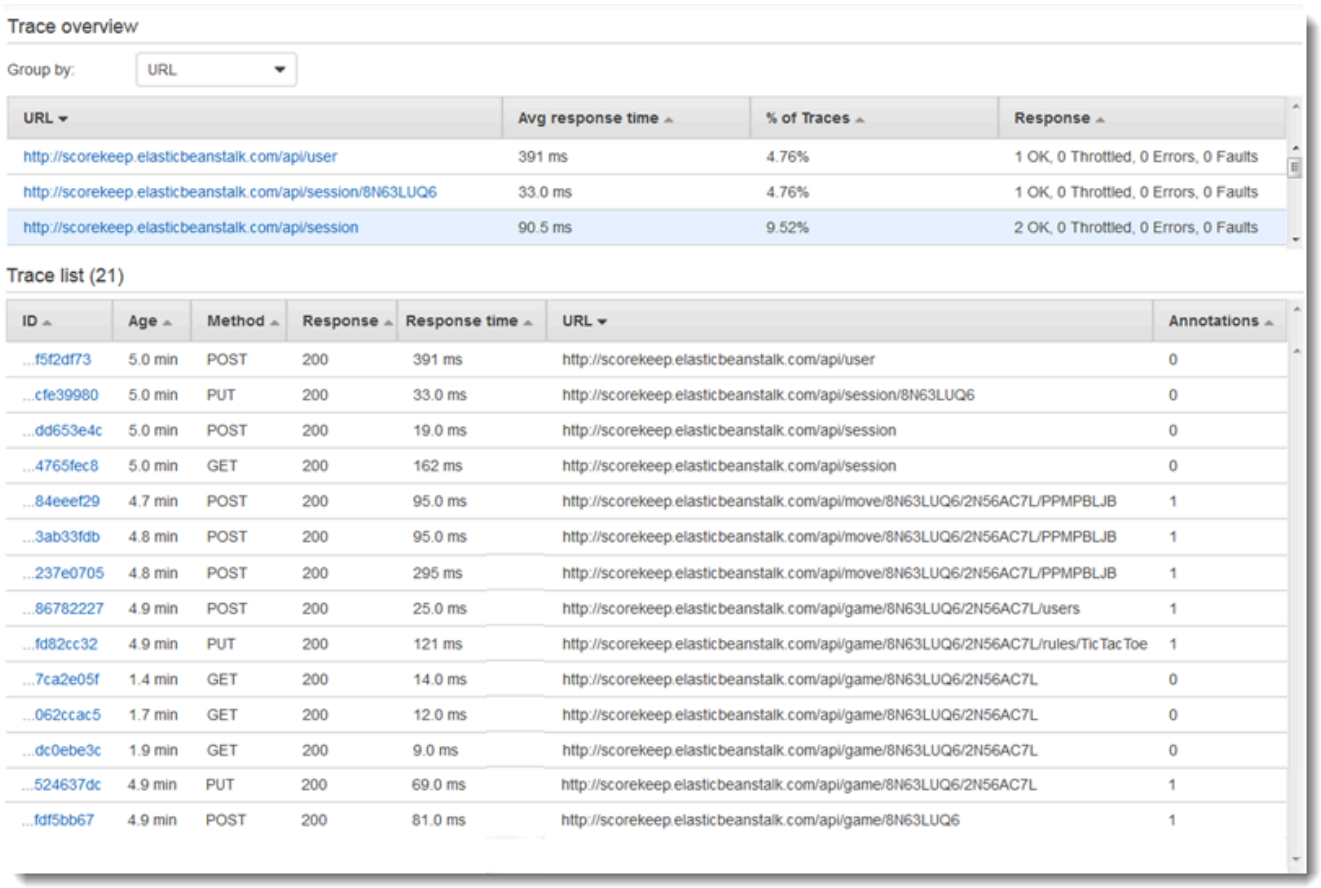
Wenn Sie über eine ereignisgesteuerte Anwendung verfügen, die Amazon SQS und Lambda verwendet, können Sie in der Trace-Map eine verknüpfte Ansicht der Traces für jede Anfrage sehen. In der Karte werden die Spuren von Nachrichtenerstellern mit den Spuren von AWS Lambda Verbrauchern verknüpft und als gestrichelte Linie dargestellt. Weitere Informationen zu ereignisgesteuerten Anwendungen finden Sie unter. Nachverfolgung ereignisgesteuerter Anwendungen

Die Seiten Traces und Trace-Details unterstützen auch die kontenübergreifende Ablaufverfolgung, bei der Traces von mehreren Konten in der Trace-Liste und in einer einzigen Trace-Map aufgeführt werden können.

X-Ray console

So zeigen Sie Spuren in der X-Ray-Konsole an

1. Öffnen Sie die Traces-Seite in der X-Ray-Konsole. Im Bereich Trace-Übersicht wird eine Liste von Traces angezeigt, die nach gemeinsamen Merkmalen wie Fehlerursachen, ResourceARN und InstanceIdgruppiert sind.
2. Um ein allgemeines Merkmal auszuwählen, um einen gruppierten Satz von Traces anzuzeigen, erweitern Sie den Abwärtspfeil neben Gruppieren nach. Die folgende Abbildung zeigt eine Trace-Übersicht der Traces, die nach der URL für gruppiert sindAWS X-Ray Musteranwendung, sowie eine Liste der zugehörigen Traces.



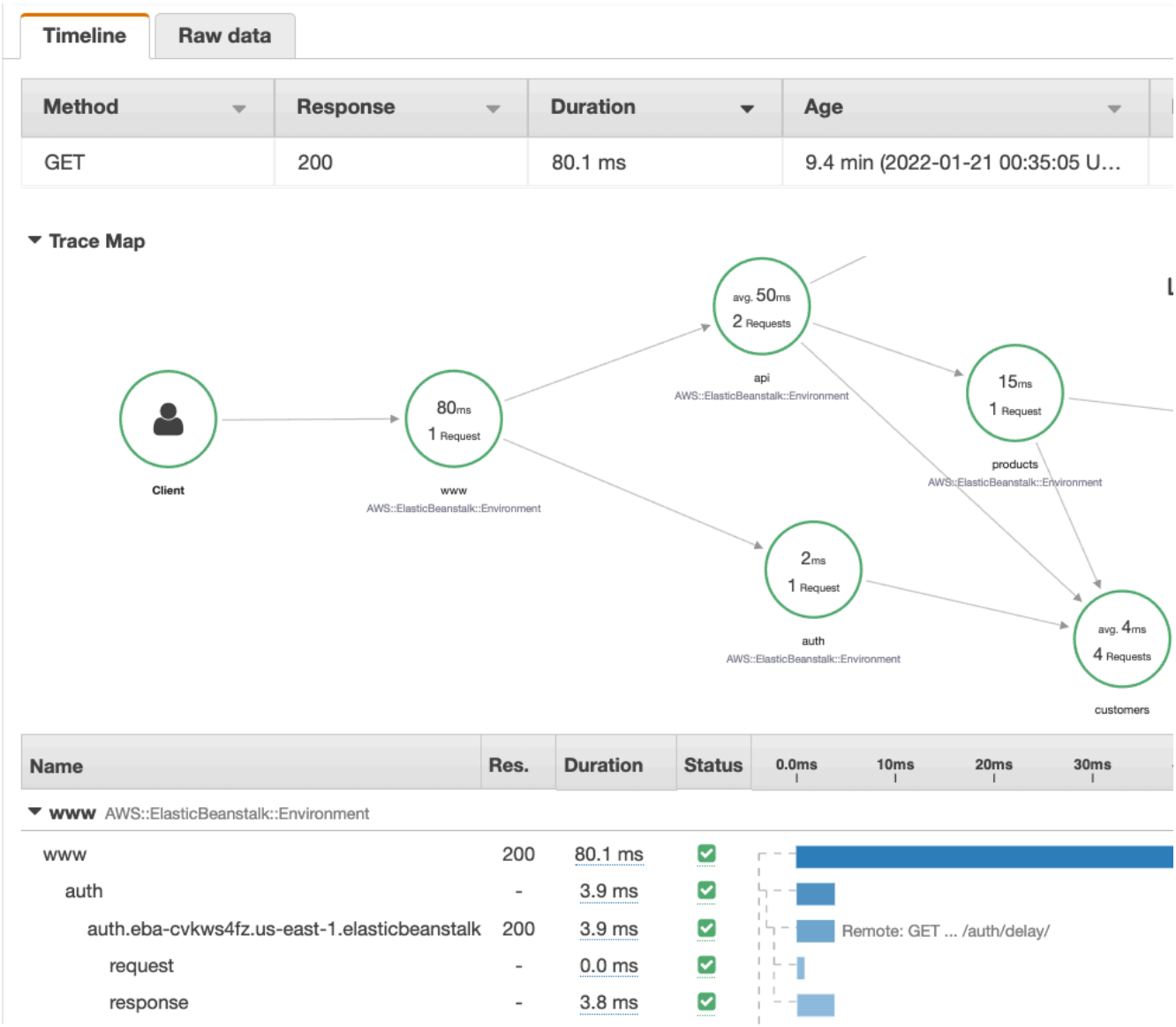
3. Wählen Sie die ID einer Ablaufverfolgung aus, um sie unter der Trace-Liste anzuzeigen. Sie können auch im Navigationsbereich die Option Service Map auswählen, um die Traces

für einen bestimmten Serviceknoten anzuzeigen. Anschließend können Sie die mit diesem Knoten verknüpften Traces anzeigen.

Auf der Registerkarte Zeitleiste wird der Anforderungsablauf für den Trace angezeigt. Er umfasst Folgendes:

- Eine Karte des Pfads für jedes Segment im Trace.
- Wie lange es gedauert hat, bis das Segment einen Knoten in der Trace-Map erreicht hat.
- Wie viele Anfragen wurden an den Knoten in der Trace-Map gestellt.

Die folgende Abbildung zeigt ein Beispiel für eine Trace-Map, die mit einer `GET` Anforderung verknüpft ist, die an eine Beispielanwendung gestellt wurde. Die Pfeile zeigen den Pfad, den jedes Segment zurückgelegt hat, um die Anfrage abzuschließen. Die Dienstknoten zeigen die Anzahl der Anfragen an, die während der `GET` Anfrage gestellt wurden.



Weitere Informationen zur Registerkarte „Zeitleiste“ finden Sie im folgenden Abschnitt „Überblick über die Ablaufverfolgung“.

Auf der Registerkarte Rohdaten werden Informationen über die Spur und die Segmente und Untersegmente, aus denen sich die Spur zusammensetzt, im JSON Format angezeigt. Diese Informationen können Folgendes beinhalten:

- Zeitstempel
- Einzigartig IDs
- Dem Segment oder Untersegment zugeordnete Ressourcen
- Die Quelle oder der Ursprung des Segments oder Untersegments

- Zusätzliche Informationen über die Anfrage an Ihre Anwendung, z. B. die Antwort auf eine HTTP-Anfrage

Erkunden Sie den Trace-Zeitplan

Der Abschnitt Zeitleiste zeigt eine Hierarchie von Segmenten und Untersegmenten neben einem horizontalen Balken, der der Zeit entspricht, die sie für die Erledigung ihrer Aufgaben aufgewendet haben. Der erste Eintrag in der Liste ist das Segment, das alle vom Service für eine einzelne Anforderung aufgezeichneten Daten darstellt. Untersegmente sind eingerückt und werden im Anschluss an das Segment aufgelistet. Spalten enthalten Informationen zu den einzelnen Segmenten.

CloudWatch console

In der CloudWatch Konsole bietet die Segment-Timeline die folgenden Informationen:

- Die erste Spalte: Listet die Segmente und Untersegmente in der ausgewählten Spur auf.
- Die Spalte Segmentstatus: Listet das Statusergebnis jedes Segments und Untersegments auf.
- Die Spalte Antwortcode: Listet einen HTTP-Antwortstatuscode auf eine Browseranfrage auf, die von dem Segment oder Untersegment gestellt wurde, sofern verfügbar.
- Die Spalte Dauer: Listet auf, wie lange das Segment oder Untersegment lief.
- Die Spalte Gehostet in: Listet den Namespace oder die Umgebung auf, in der das Segment oder Untersegment ausgeführt wird, falls zutreffend. Weitere Informationen finden Sie unter Gesammelte Dimensionen und Dimensionskombinationen.
- Die letzte Spalte: Zeigt horizontale Balken an, die der Dauer entsprechen, in der das Segment oder Untersegment im Verhältnis zu den anderen Segmenten oder Untersegmenten in der Zeitleiste ausgeführt wurde.

Um die Liste der Segmente und Untersegmente nach Serviceknoten zu gruppieren, aktivieren Sie die Option Nach Knoten gruppieren.

X-Ray console

Wählen Sie auf der Seite mit den Trace-Details die Registerkarte Timeline aus, um die Timeline für jedes Segment und Untersegment anzuzeigen, aus denen ein Trace besteht.

In der X-Ray-Konsole bietet die Timeline die folgenden Informationen:

- Die Spalte Name: Listet die Namen der Segmente und Untersegmente im Trace auf.
- Die Spalte Res.: Listet einen HTTP-Antwortstatuscode auf eine Browseranfrage auf, die von dem Segment oder Untersegment gestellt wurde, sofern verfügbar.
- Die Spalte Dauer: Listet auf, wie lange das Segment oder Untersegment lief.
- Die Spalte Status: Listet das Ergebnis des Status eines Segments oder Untersegments auf.
- Die letzte Spalte: Zeigt horizontale Balken an, die der Dauer entsprechen, in der das Segment oder Untersegment im Verhältnis zu den anderen Segmenten oder Untersegmenten in der Zeitleiste ausgeführt wurde.

Um die rohen Trace-Daten anzuzeigen, die die Konsole zur Generierung der Zeitleiste verwendet, wählen Sie die Registerkarte Rohdaten. Die Rohdaten zeigen Ihnen Informationen über den Trace und die Segmente und Untersegmente, aus denen sich der Trace zusammensetzt, im `JSON` Format. Diese Informationen können Folgendes beinhalten:

- Zeitstempel
- Einzigartig IDs
- Dem Segment oder Untersegment zugeordnete Ressourcen
- Die Quelle oder der Ursprung des Segments oder Untersegments
- Zusätzliche Informationen über die Anfrage an Ihre Anwendung, z. B. die Antwort auf eine HTTP-Anfrage.

Wenn Sie ein instrumentiertes AWS SDK verwenden, HTTP, oder SQL Client für Aufrufe externer Ressourcen, das X-Ray SDK zeichnet Untersegmente automatisch auf. Sie können das X-Ray SDK auch verwenden, um benutzerdefinierte Untersegmente für jede Funktion oder jeden Codeblock aufzuzeichnen. Zusätzliche Untersegmente, die aufgezeichnet werden, während ein benutzerdefiniertes Untersegment geöffnet ist, werden zu untergeordneten Segmenten des benutzerdefinierten Untersegments.

Anzeigen von Segmentdetails

Wählen Sie in der Trace-Zeitleiste den Namen eines Segments aus, um dessen Details anzuzeigen.

Im Bereich „Segmentdetails“ werden die Registerkarten „Übersicht“, „Ressourcen“, „Anmerkungen“, „Metadaten“, „Ausnahmen“ und „SQL“ angezeigt. Es gelten die folgenden Bedingungen:

- Die Registerkarte Overview (Übersicht) zeigt Informationen zu Anforderung und Antwort an. Zu den Informationen gehören der Name, die Startzeit, die Endzeit, die Anforderungs-URL, der Anforderungsvorgang, der Anforderungsantwortcode sowie etwaige Fehler und Störungen.
- Auf der Registerkarte Ressourcen für ein Segment werden Informationen aus dem X-Ray SDK und zu den AWS Ressourcen angezeigt, auf denen Ihre Anwendung ausgeführt wird. Verwenden Sie die Amazon EC2 - AWS Elastic Beanstalk oder Amazon ECS-Plug-ins für das X-Ray SDK, um servicespezifische Ressourceninformationen aufzuzeichnen. Weitere Informationen zu Plug-ins finden Sie im Konfiguration des X-Ray SDK for Java Abschnitt Service-Plugins unter.
- Auf den verbleibenden Registerkarten werden Anmerkungen, Metadaten und Ausnahmen angezeigt, die für das Segment aufgezeichnet wurden. Ausnahmen werden automatisch erfasst, wenn sie aus einer instrumentierten Anfrage generiert werden. Anmerkungen und Metadaten enthalten zusätzliche Informationen, die Sie mithilfe der vom X-Ray SDK bereitgestellten Operationen aufzeichnen. Verwenden Sie das X-Ray SDK, um Anmerkungen oder Metadaten zu Ihren Segmenten hinzuzufügen. Weitere Informationen finden Sie unter dem sprachspezifischen Link, der unter Instrumentierung Ihrer Anwendung mit in aufgeführt ist. AWS X-Ray SDKs Instrumentierung Ihrer Anwendung für AWS X-Ray

Anzeigen von Untersegmentdetails

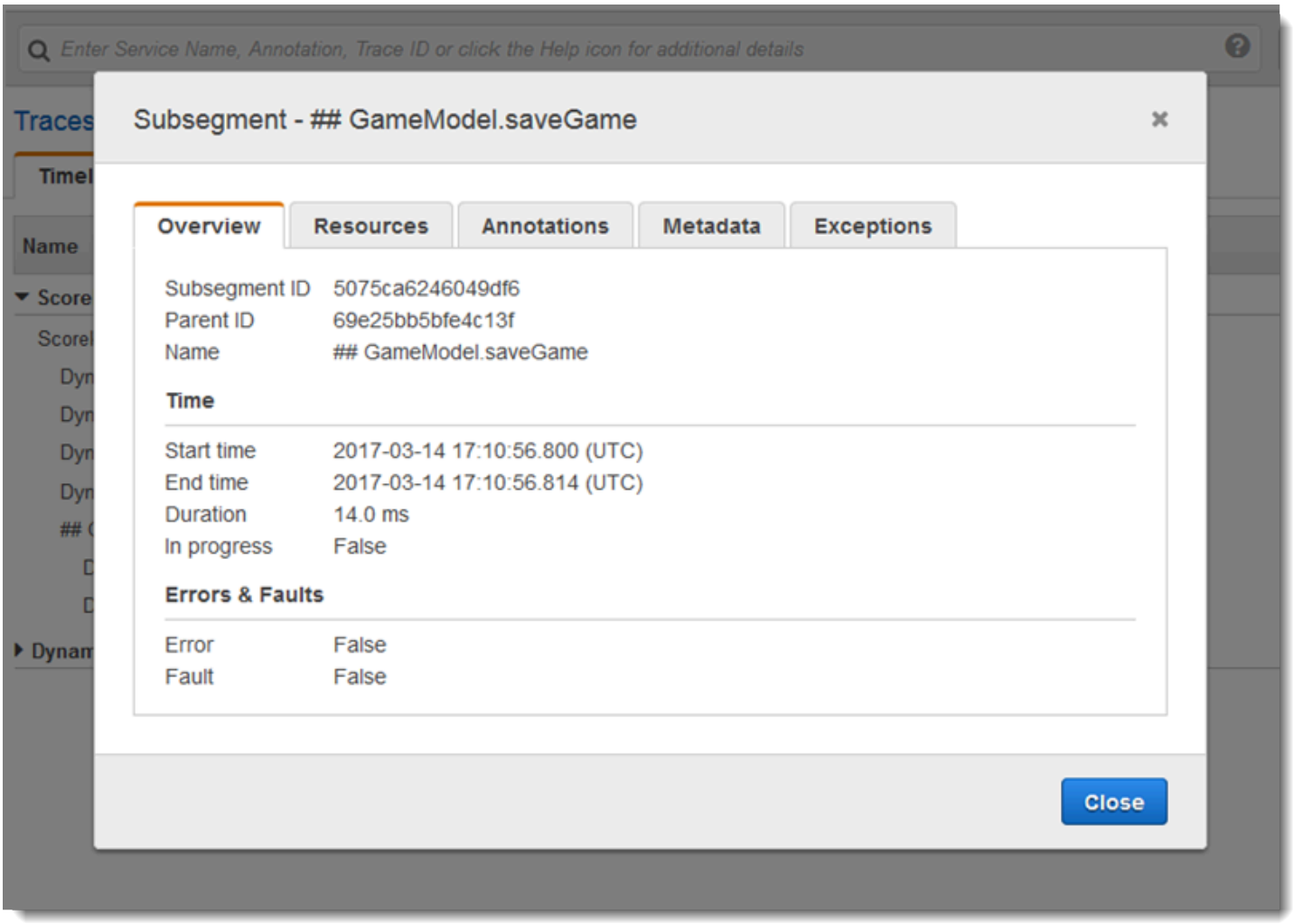
Wählen Sie in der Ablaufverfolgungszeitleiste den Namen eines Untersegments aus, um dessen Details anzuzeigen:

- Die Registerkarte „Übersicht“ enthält Informationen über die Anfrage und die Antwort. Dazu gehören der Name, die Startzeit, die Endzeit, die Dauer und die Anfrage URL, Vorgang der Anfrage, Antwortcode der Anfrage und etwaige Fehler und Störungen. Für die mit instrumentierten Clients generierten Untersegmente enthält die Registerkarte Overview (Übersicht) Informationen zu Anforderung und Antwort aus der Sicht Ihrer Anwendung.
- Auf der Registerkarte Ressourcen für ein Untersegment werden Details zu den AWS Ressourcen angezeigt, die zur Ausführung des Untersegments verwendet wurden. Die Registerkarte Ressourcen kann beispielsweise einen AWS Lambda Funktions-ARN, Informationen über eine DynamoDB-Tabelle, jede aufgerufene Operation und eine Anforderungs-ID enthalten.
- Auf den verbleibenden Registerkarten werden Anmerkungen, Metadaten und Ausnahmen angezeigt, die für das Untersegment aufgezeichnet wurden. Ausnahmen werden automatisch erfasst, wenn sie aus einer instrumentierten Anfrage generiert werden. Anmerkungen und Metadaten enthalten zusätzliche Informationen, die Sie mithilfe der vom X-Ray SDK bereitgestellten Operationen aufzeichnen. Verwenden Sie das X-Ray SDK, um Anmerkungen

oder Metadaten zu Ihren Segmenten hinzuzufügen. Weitere Informationen finden Sie unter dem sprachspezifischen Link, der unter Instrumentierung Ihrer Anwendung mit in aufgeführt ist. AWS X-Ray SDKs Instrumentierung Ihrer Anwendung für AWS X-Ray

Bei benutzerdefinierten Untersegmenten zeigt die Registerkarte Overview (Übersicht) den Namen des Untersegments an, den Sie angeben können, um den Code-Bereich oder die Funktion anzugeben, der oder die aufgezeichnet wird. Weitere Informationen finden Sie unter dem sprachspezifischen Link, der unter Instrumentierung Ihrer Anwendung mit in aufgeführt ist. AWS X-Ray SDKs Generieren von benutzerdefinierten Untersegmenten mit dem X-Ray SDK for Java

Die folgende Abbildung zeigt die Registerkarte „Übersicht“ für ein benutzerdefiniertes Untersegment. Die Übersicht enthält die Untersegment-ID, die übergeordnete ID, den Namen, die Start- und Endzeiten, die Dauer, den Status und die Fehler oder Störungen.



Die Registerkarte „Metadaten“ für ein benutzerdefiniertes Untersegment enthält Informationen in JSON Format über die Ressourcen, die von diesem Untersegment verwendet werden.

Verwenden von Filterausdrücken

Verwenden Sie Filterausdrücke, um eine Trace-Map oder Traces für eine bestimmte Anfrage, einen bestimmten Dienst, eine Verbindung zwischen zwei Diensten (ein Edge) oder Anfragen, die eine Bedingung erfüllen, anzuzeigen. X-Ray bietet eine Sprache für Filterausdrücke zum Filtern von Anfragen, Diensten und Kanten auf der Grundlage von Daten in Anforderungsheadern, Antwortstatus und indizierten Feldern in den ursprünglichen Segmenten.

Wenn Sie einen Zeitraum für die Anzeige von Traces in der X-Ray-Konsole auswählen, erhalten Sie möglicherweise mehr Ergebnisse, als die Konsole anzeigen kann. In der rechten oberen Ecke der Konsole wird die Anzahl der gescannten Ablaufverfolgungen angezeigt und Sie können ablesen, ob weitere Ablaufverfolgungen verfügbar sind. Sie können einen Filterausdruck verwenden, um die Ergebnisse auf die Spuren einzugrenzen, nach denen Sie suchen möchten.

Themen

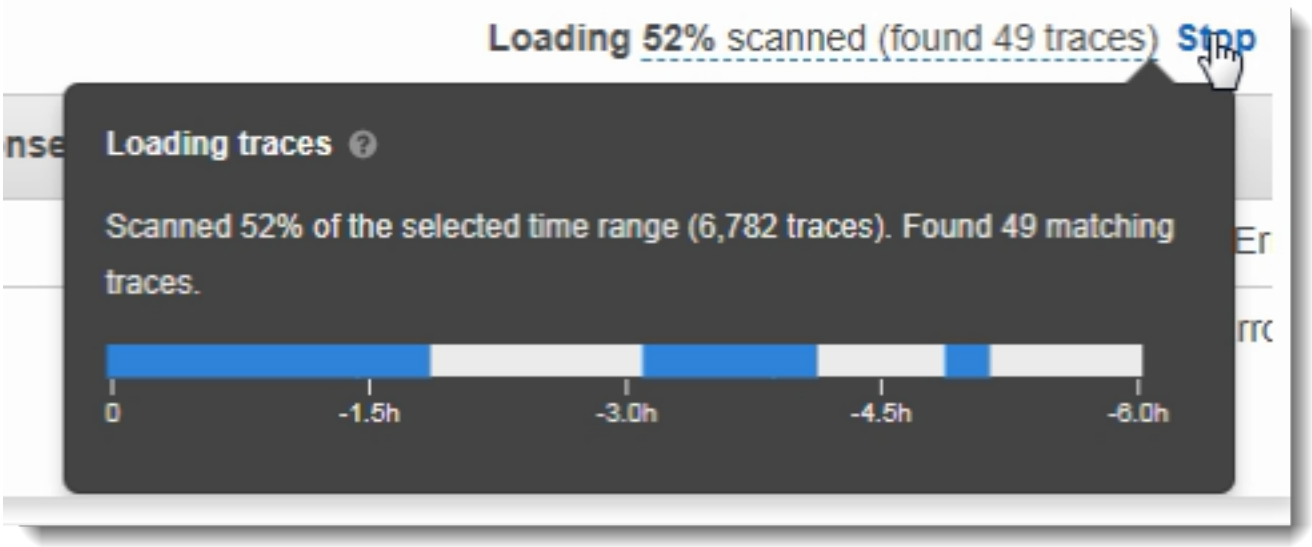
Details zu Filterausdrücken

Wenn Sie einen Knoten in der Trace-Map auswählen, erstellt die Konsole einen Filterausdruck, der auf dem Dienstnamen des Knotens und den auf Ihrer Auswahl beruhenden Fehlertypen basiert. Um Ablaufverfolgungen zu finden, die Leistungsprobleme zeigen oder zu bestimmten Anfragen gehören, können Sie den von der Konsole bereitgestellten Ausdruck anpassen oder einen eigenen erstellen. Wenn Sie Anmerkungen mit dem X-Ray SDK hinzufügen, können Sie auch nach dem Vorhandensein eines Annotationsschlüssels oder dem Wert eines Schlüssels filtern.

> **Note**
>
> Wenn Sie in der Trace-Map einen relativen Zeitraum und einen Knoten auswählen, konvertiert die Konsole den Zeitraum in eine absolute Start- und Endzeit. Um sicherzustellen, dass die Ablaufverfolgungen für den Knoten in den Suchergebnissen angezeigt werden, und um das Scannen von Zeiträumen zu vermeiden, in denen der Knoten nicht aktiv war, werden nur Zeiten in den Zeitraum einbezogen, zu denen der Knoten Ablaufverfolgungen gesendet hat. Um relativ zur aktuellen Zeit zu suchen, können Sie wieder zum relativen Zeitraum auf der Seite der Ablaufverfolgungen wechseln und einen erneuten Scan durchführen.

Wenn immer noch mehr Ergebnisse vorliegen, als in der Konsole angezeigt werden können, gibt die Konsole die Anzahl der Ablaufverfolgungen an, die die Kriterien erfüllt haben, sowie die Anzahl der gescannten Ablaufverfolgungen. Der angezeigte Prozentsatz entspricht dem Prozentsatz des ausgewählten Zeitraums, der gescannt wurde. Um sicherzustellen, dass alle passenden Ablaufverfolgungen in den Ergebnissen angezeigt werden, grenzen Sie Ihren Filterausdruck weiter ein oder wählen einen kürzeren Zeitraum aus.

Die Konsole beginnt den Scan am Ende des Zeitraums und arbeitet sich nach vorne durch, um Ihnen die aktuellsten Ergebnisse zuerst anzuzeigen. Wenn viele Ablaufverfolgungen vorliegen, aber nur wenige Ergebnisse, teilt die Konsole den Zeitraum in Abschnitte ein und scannt diese parallel. Der Fortschrittsbalken zeigt die Abschnitte des Zeitraums an, die gescannt wurden.



Verwenden von Filterausdrücken mit Gruppen

Gruppen sind eine Sammlung von Ablaufverfolgungen, die durch einen Filterausdruck definiert sind. Sie können Gruppen verwenden, um zusätzliche Servicegraphen zu erstellen und CloudWatch Amazon-Metriken bereitzustellen.

Gruppen werden über ihren Namen oder einen Amazon-Ressourcennamen (ARN) identifiziert und enthalten einen Filterausdruck. Der Service vergleicht eingehende Ablaufverfolgungen mit dem Ausdruck und speichert sie entsprechend.

Sie können Gruppen mit dem Dropdown-Menü links neben der Suchleiste des Filterausdrucks erstellen und ändern.

> **Note**
>
> Wenn der Service einen Fehler beim Qualifizieren einer Gruppe feststellt, ist diese Gruppe nicht mehr in der Verarbeitung eingehender Ablaufverfolgungen enthalten und es wird eine Fehlermetrik aufgezeichnet.

Weitere Informationen zu Gruppen finden Sie unterGruppen konfigurieren.

Syntax für Filterausdrücke

Filterausdrücke können ein Schlüsselwort, einen unären oder binären Operator und einen Wert für den Vergleich enthalten.

```
keyword operator value
```

Für verschiedene Arten von Schlüsselwörtern sich unterschiedliche Operatoren erhältlich. Beispielsweise ist `responsetime` ein Zahlenschlüsselwort, das mit auf Zahlen bezüglichen Operatoren verglichen werden kann.

Example— Anfragen, bei denen die Antwortzeit mehr als 5 Sekunden betrug

```
responsetime > 5
```

Sie können mehrere Ausdrücke zu einem zusammengesetzten Ausdruck verbinden, indem Sie die Operatoren `AND` oder `OR` verwenden.

Example— Anfragen mit einer Gesamtdauer von 5—8 Sekunden

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

Einfache Schlüsselwörter und Operatoren finden Probleme nur auf der Ebene der Ablaufverfolgungen. Wenn ein Fehler dahinter auftritt, von Ihrer Anwendung jedoch berücksichtigt und nicht an den Benutzer zurückgegeben wird, findet eine Suche nach `error` diesen Fehler nicht.

Zum Finden von Ablaufverfolgungen mit nachgeordneten Fehlern können Sie komplexe Schlüsselwörter `service()` und `edge()` verwenden. Diese Schlüsselwörter ermöglichen die Anwendung eines Filterausdrucks auf alle nachgeordneten Knoten, einen einzelnen nachgeordneten Knoten oder eine Edge zwischen zwei Knoten. Für mehr Granularität können Sie Services und Edges mit der id()-Funktion nach Typ filtern.

Boolesche Schlüsselwörter

Boolesche Schlüsselwortwerte sind entweder „true“ oder „false“. Verwenden Sie diese Schlüsselwörter, um Ablaufverfolgungen zu finden, die zu Fehlern geführt haben.

Boolesche Schlüsselwörter

- `ok`— Der Statuscode der Antwort lautete 2XX Success.
- `error`— Der Antwortstatuscode lautete 4XX Client Error.
- `throttle`— Der Antwortstatuscode lautete 429 Too Many Requests.
- `fault`— Der Antwortstatuscode lautete 5XX Server Error.
- `partial`— Die Anfrage enthält unvollständige Segmente.
- `inferred`— Die Anfrage hat abgeleitete Segmente.
- `first`— Element ist das erste Element einer aufgezählten Liste.
- `last`— Element ist das letzte Element einer Aufzählungsliste.
- `remote`— Die Ursachenentität ist entfernt.
- `root`— Service ist der Einstiegspunkt oder das Stammsegment einer Ablaufverfolgung.

Boolesche Operatoren finden Segmente, bei denen der angegebene Schlüssel `true` oder `false` ist.

Boolesche Operatoren

- none — Der Ausdruck ist wahr, wenn das Schlüsselwort wahr ist.

- `!` — Der Ausdruck ist wahr, wenn das Schlüsselwort falsch ist.
- `=`, `!=` — Vergleicht den Wert des Schlüsselworts mit der Zeichenfolge `true` oder`false`. Diese Operatoren verhalten sich genauso wie die anderen Operatoren, sind jedoch expliziter.

Example— Der Antwortstatus ist 2XX OK

```
ok
```

Example— der Antwortstatus ist nicht 2XX OK

```
!ok
```

Example— der Antwortstatus ist nicht 2XX OK

```
ok = false
```

Example— Der letzte aufgezählte Fehler-Trace hat den Fehlernamen „deserialize“

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

Example— Anfragen mit entfernten Segmenten, bei denen die Abdeckung größer als 0,7 ist und der Dienstname „traces“ lautet

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

Example— Anfragen mit abgeleiteten Segmenten, bei denen der Diensttyp „aws:DynamoDB“ ist

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

Example— Anfragen, die ein Segment mit dem Namen „Datenebene“ als Stamm haben

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

Zahlenschlüsselwörter

Verwenden Sie Zahlenschlüsselwörter, um nach Anforderungen mit einer bestimmten Reaktionszeit, Dauer oder einem bestimmten Reaktionsstatus zu suchen.

Zahlenschlüsselwörter

- `responsetime`— Zeit, die der Server benötigt hat, um eine Antwort zu senden.
- `duration`— Gesamtdauer der Anfrage, einschließlich aller Downstream-Aufrufe.
- `http.status`— Statuscode der Antwort.
- `index`— Position eines Elements in einer aufgezählten Liste.
- `coverage`— Dezimaler Prozentsatz der Antwortzeit der Entität im Vergleich zur Antwortzeit des Stammsegments. Gilt nur für die Reaktionszeit der Ursachenentitäten.

Zahlenoperatoren

Zahlenschlüsselwörter verwenden Standard-Operatoren für Gleichheit und Vergleich.

- =, `!`= — Das Schlüsselwort ist gleich oder ungleich einem Zahlenwert.
- <,<=,>, >= — Das Schlüsselwort ist kleiner oder größer als ein Zahlenwert.

Example— Der Antwortstatus ist nicht 200 OK

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

Example— Anfrage, bei der die Gesamtdauer 5—8 Sekunden betrug

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

Example— Anfragen, die in weniger als 3 Sekunden erfolgreich abgeschlossen wurden, einschließlich aller Downstream-Aufrufe

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

Example— Entität mit einer aufgezählten Liste, deren Index größer als 5 ist

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

Example— Anfragen, bei denen die letzte Entität, deren Abdeckung größer als 0,8 ist

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

Zeichenfolgen-Schlüsselwörter

Verwenden Sie String-Schlüsselwörter, um Traces mit einem bestimmten Text in den Headern der Anfrage oder nach einem bestimmten Benutzer IDs zu finden.

Zeichenfolgen-Schlüsselwörter

- `http.url`— URL anfordern.
- `http.method`— Methode anfordern.
- `http.useragent`— User-Agent-Zeichenfolge anfordern.
- `http.clientip`— Die IP-Adresse des Anforderers.
- `user`— Wert des Benutzerfeldes in einem beliebigen Segment im Trace.
- `name`— Der Name eines Dienstes oder einer Ausnahme.
- `type`— Art der Dienstleistung.
- `message`— Ausnahmemeldung.
- `availabilityzone`— Wert des AvailabilityZone-Felds für ein beliebiges Segment im Trace.
- `instance.id`— Wert des Instanz-ID-Felds in einem beliebigen Segment im Trace.
- `resource.arn`— Wert des Ressourcen-ARN-Felds für ein beliebiges Segment im Trace.

Zeichenfolgenoperatoren finden Werte, die identisch mit bestimmtem Text sind oder diesen enthalten. Werte müssen stets in Anführungszeichen angegeben werden.

Zeichenfolgenoperatoren

- `=`, `!=` — Das Schlüsselwort ist gleich oder ungleich einem Zahlenwert.
- `CONTAINS`— Das Schlüsselwort enthält eine bestimmte Zeichenfolge.
- `BEGINSWITH`, `ENDSWITH` — Das Schlüsselwort beginnt oder endet mit einer bestimmten Zeichenfolge.

Example— http.url-Filter

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

Um zu testen, ob ein Feld in einer Ablaufverfolgung vorhanden ist (unabhängig von seinem Wert), prüfen Sie, ob es die leere Zeichenfolge enthält.

Example— Benutzerfilter

Finde alle Spuren mit dem Benutzer IDs.

```
user CONTAINS ""
```

Example— Wählen Sie Traces mit einer Fehlerursache aus, zu der auch ein Dienst mit dem Namen „Auth“ gehört

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

Example— wählt Traces mit einer Hauptursache für die Antwortzeit aus, deren letzter Service den Typ DynamoDB hat

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

Example— wählt Traces mit einer Fehlerursache aus, deren letzte Ausnahme die Meldung „Zugriff verweigert für account_id: 1234567890" enthält

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

Komplexe Schlüsselwörter

Verwenden Sie komplexe Schlüsselwörter, um Anfragen anhand des Servicenamens, des Edge-Namens oder des Anmerkungswerts zu finden. Für Services und Edges können Sie einen zusätzlichen Filterausdruck angeben, der sich auf den Service oder die Edge bezieht. Für Anmerkungen können Sie nach dem Wert einer Anmerkung mit einem bestimmten Schlüssel filtern und dafür boolesche Werte, Zahlen oder Zeichenfolgenoperatoren verwenden.

Komplexe Schlüsselwörter

- `annotation[`*`key`*`]`— Wert einer Anmerkung mit Feld. *`key`* Der Wert einer Anmerkung kann ein boolescher Wert, eine Zahl oder eine Zeichenfolge sein, sodass Sie jeden der Vergleichsoperatoren dieser Typen verwenden können. Sie können dieses Schlüsselwort in Kombination mit dem `edge` Schlüsselwort `service` or verwenden. Ein Kommentarschlüssel, der Punkte (Punkte) enthält, muss in eckige Klammern ([]) eingeschlossen werden.

- `edge(source, destination) {filter}`— Verbindung zwischen Diensten *source* und*destination*. Optionale geschweifte Klammern können einen Filterausdruck enthalten, der für Segmente in dieser Verbindung gilt.
- `group.name / group.arn`— Der Wert des Filterausdrucks einer Gruppe, auf den durch den Gruppennamen oder Gruppen-ARN verwiesen wird.
- `json`— JSON-Ursachenobjekt. Schritte zum programmgesteuerten Erstellen von JSON-Entitäten finden Sie unter Daten aus AWS X-Ray abrufen.
- `service(name) {filter}`— Dienst mit Namen. *name* Optionale geschweifte Klammern können einen Filterausdruck enthalten, der für vom Service erstellte Segmente gilt.

Verwenden Sie das Schlüsselwort service, um Traces für Anfragen zu finden, die einen bestimmten Knoten auf Ihrer Trace-Map erreichen.

Komplexe Schlüsselwort-Operatoren finden Segmente, in denen der angegebene Schlüssel gesetzt wurde oder nicht.

Komplexe Schlüsselwort-Operatoren

- none — Der Ausdruck ist wahr, wenn das Schlüsselwort gesetzt ist. Wenn das Schlüsselwort vom Typ Boolean ist, wird es als boolescher Wert ausgewertet.
- `!`— Der Ausdruck ist wahr, wenn das Schlüsselwort nicht gesetzt ist. Wenn das Schlüsselwort vom Typ Boolean ist, wird es als boolescher Wert ausgewertet.
- =, `!=` — Vergleicht den Wert des Schlüsselworts.
- `edge(source, destination) {filter}`— Verbindung zwischen Diensten *source* und*destination*. Optionale geschweifte Klammern können einen Filterausdruck enthalten, der für Segmente in dieser Verbindung gilt.
- `annotation[key]`— Wert einer Anmerkung mit Feld*key*. Der Wert einer Anmerkung kann ein boolescher Wert, eine Zahl oder eine Zeichenfolge sein, sodass Sie jeden der Vergleichsoperatoren dieser Typen verwenden können. Sie können dieses Schlüsselwort in Kombination mit dem `edge` Schlüsselwort `service` or verwenden.
- `json`— JSON-Ursachenobjekt. Schritte zum programmgesteuerten Erstellen von JSON-Entitäten finden Sie unter Daten aus AWS X-Ray abrufen.

Verwenden Sie das Schlüsselwort service, um Traces für Anfragen zu finden, die einen bestimmten Knoten auf Ihrer Trace-Map erreichen.

Example— Servicefilter

Anforderungen mit einem Aufruf an `api.example.com` mit einem Fehler (Fehler der Serie 500).

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

Sie können den Servicenahmen ausschließen, um einen Filterausdruck auf alle Knoten in Ihrer Service-Übersicht anzuwenden.

Example— Servicefilter

Anfragen, die an einer beliebigen Stelle auf Ihrer Trace-Map einen Fehler verursacht haben.

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

Das Edge-Schlüsselwort wendet einen Filterausdruck auf eine Verbindung zwischen zwei Knoten an.

Example— Kantenfilter

Anforderung, bei der der Service `api.example.com` einen Aufruf an `backend.example.com` vorgenommen hat, der mit einem Fehler fehlgeschlagen ist.

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

Sie können auch den `!`-Operator mit Service- und Edge-Schlüsselwörtern verwenden, um einen Service oder eine Edge aus den Ergebnissen eines anderen Filterausdrucks auszuschließen.

Example— Service- und Anfragefilter

Anforderung, bei der die URL mit `http://api.example.com/` beginnt und `/v2/` enthält, aber kein Service mit dem Namen `api.example.com` erreicht wird.

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

Example— Service- und Antwortzeitfilter

Findet Spuren, bei denen der Wert eingestellt `http url` ist und die Reaktionszeit länger als 2 Sekunden ist.

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

Für Anmerkungen können Sie alle Traces aufrufen, bei denen der Wert gesetzt `annotation[key]` ist, oder die Vergleichsoperatoren verwenden, die dem Wertetyp entsprechen.

Example— Anmerkung mit Zeichenkettenwert

Anforderungen mit einer Anmerkung mit dem Namen `gameid` und dem Zeichenfolgenwert `"817DL6V0"`.

```
annotation[gameid] = "817DL6V0"
```

Example— Anmerkung ist gesetzt

Anfragen mit einer Anmerkung namens `age` set.

```
annotation[age]
```

Example— Anmerkung ist nicht gesetzt

Anfragen ohne eine Anmerkung namens `age` set.

```
!annotation[age]
```

Example— Anmerkung mit Zahlenwert

Anforderungen mit einer Anmerkung mit einem numerischen Wert größer als 29.

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

Example— Anmerkung in Kombination mit Service oder Edge

```
service { annotation[request.id] = "917DL6V0" }
```

```
edge { source.annotation[request.id] = "916DL6V0" }
```

```
edge { destination.annotation[request.id] = "918DL6V0" }
```

Example— Gruppe mit Benutzer

Anfragen, bei denen Traces auf den `high_response_time` Gruppenfilter treffen (z. B.`responseTime > 3`) und der Benutzer den Namen Alice hat.

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

Example— JSON mit der Ursachenentität

Anforderungen mit übereinstimmenden Ursachenentitäten.

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

id-Funktion

Wenn Sie einen Service-Namen für die Schlüsselwörter `service` oder `edge` bereitstellen, erhalten Sie Ergebnisse für alle Knoten mit diesem Namen. Für eine präzisere Filterung können Sie mit der `id`-Funktion zusätzlich zu einem Namen einen Servicetyp angeben, um zwischen Knoten mit demselben Namen zu differenzieren.

Verwenden Sie die `account.id` Funktion, um ein bestimmtes Konto für den Dienst anzugeben, wenn Sie Traces von mehreren Konten in einem Monitoring-Konto anzeigen.

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

Sie können die `id`-Funktion anstelle eines Service-Namens in Service- und Edge-Filtern verwenden.

```
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 }
```

AWS Lambda Funktionen führen beispielsweise zu zwei Knoten in der Trace-Map; einer für den Funktionsaufruf und einer für den Lambda-Service. Die zwei Knoten haben denselben Namen, aber unterschiedliche Arten. Ein Standard-Servicefilter findet Ablaufnachverfolgungen für beide.

Example— Dienstfilter

Anforderungen, die einen Fehler bei einem Service mit dem Namen `random-name` enthalten.

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

Verwenden Sie die `id`-Funktion zur Eingrenzung der Suche auf Fehler bei der Funktion selbst, unter Ausschluss von Fehlern vom Service.

Example— Servicefilter mit ID-Funktion

Anforderungen, die einen Fehler bei einem Service mit dem Namen `random-name` und dem Typ `AWS::Lambda::Function` enthalten.

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

Um Knoten nach Typ zu suchen, können Sie den Namen auch vollständig ausschließen.

Example— Servicefilter mit ID-Funktion und Servicetyp

Anforderungen, die einen Fehler bei einem Service des Typs `AWS::Lambda::Function` enthalten.

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

Um nach Knoten für einen bestimmten Knoten zu suchen AWS-Konto, geben Sie eine Konto-ID an.

Example— Servicefilter mit ID-Funktion und Konto-ID

Anfragen, die einen Dienst innerhalb einer bestimmten Konto-ID beinhaltenAWS::Lambda::Function.

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

Kontenübergreifende Rückverfolgung

AWS X-Ray unterstützt kontenübergreifende Beobachtbarkeit, sodass Sie Anwendungen überwachen und Fehler beheben können, die sich über mehrere Konten innerhalb eines. AWS-Region Sie können Ihre Metriken, Protokolle und Traces in allen verknüpften Konten nahtlos suchen, visualisieren und analysieren, als ob Sie mit einem einzigen Konto arbeiten würden. Auf diese Weise erhalten Sie einen vollständigen Überblick über Anfragen, die über mehrere Konten hinweg eingehen. Sie können kontenübergreifende Traces in der X-Ray-Trace-Map und auf den Traces-Seiten in der CloudWatchKonsole anzeigen.

Die gemeinsam genutzten Observability-Daten können jede der folgenden Telemetriearten beinhalten:

- Metriken bei Amazon CloudWatch

- Gruppen in Amazon CloudWatch Logs protokollieren
- Spuren in AWS X-Ray
- Anwendungen in Amazon CloudWatch Application Insights

Konfigurieren Sie die kontoübergreifende Beobachtbarkeit

Um die kontenübergreifende Beobachtbarkeit zu aktivieren, richten Sie ein oder mehrere AWS Überwachungskonten ein und verknüpfen Sie sie mit mehreren Quellkonten. Ein Monitoring-Konto ist ein zentrales System AWS-Konto , das Beobachtungsdaten, die aus Quellkonten generiert wurden, einsehen und mit ihnen interagieren kann. Ein Quellkonto ist eine Person AWS-Konto , die Beobachtbarkeitsdaten für die darin enthaltenen Ressourcen generiert.

Quellkonten teilen ihre Beobachtbarkeitsdaten mit Monitoringkonten. Traces werden von jedem Quellkonto auf bis zu fünf Überwachungskonten kopiert. Kopien der Traces von den Quellkonten auf das erste Überwachungskonto sind kostenlos. Kopien von Traces, die an zusätzliche Überwachungskonten gesendet werden, werden jedem Quellkonto auf der Grundlage der Standardpreise in Rechnung gestellt. Weitere Informationen finden Sie unter AWS X-Ray Preise und CloudWatch Amazon-Preise.

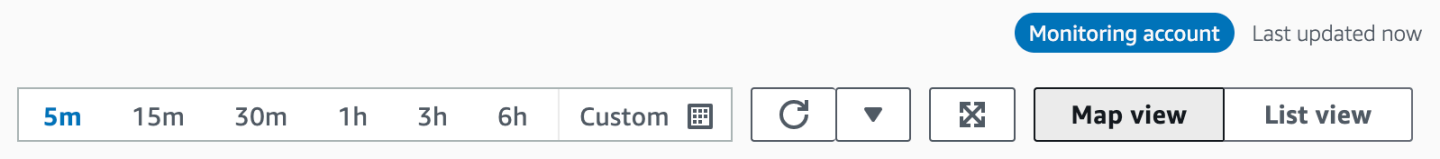
Verwenden Sie die CloudWatch Konsole oder die neuen Observability Access Manager-Befehle in der AND-API, um Verknüpfungen zwischen Monitoring-Konten AWS CLI und Quellkonten zu erstellen. Weitere Informationen finden Sie unter Kontoübergreifende Beobachtbarkeit von CloudWatch .

> **Note**
>
> Röntgenspuren werden dort abgerechnet, AWS-Konto wo sie empfangen wurden. Wenn sich eine Stichprobenanfrage über mehrere Dienste erstreckt AWS-Konto, zeichnet jedes Konto eine separate Ablaufverfolgung auf, und alle Traces verwenden dieselbe Trace-ID. Weitere Informationen zu kontenübergreifenden Observability-Preisen finden Sie unter Preise und AWS X-Ray CloudWatch Amazon-Preise.

Kontoübergreifende Traces anzeigen

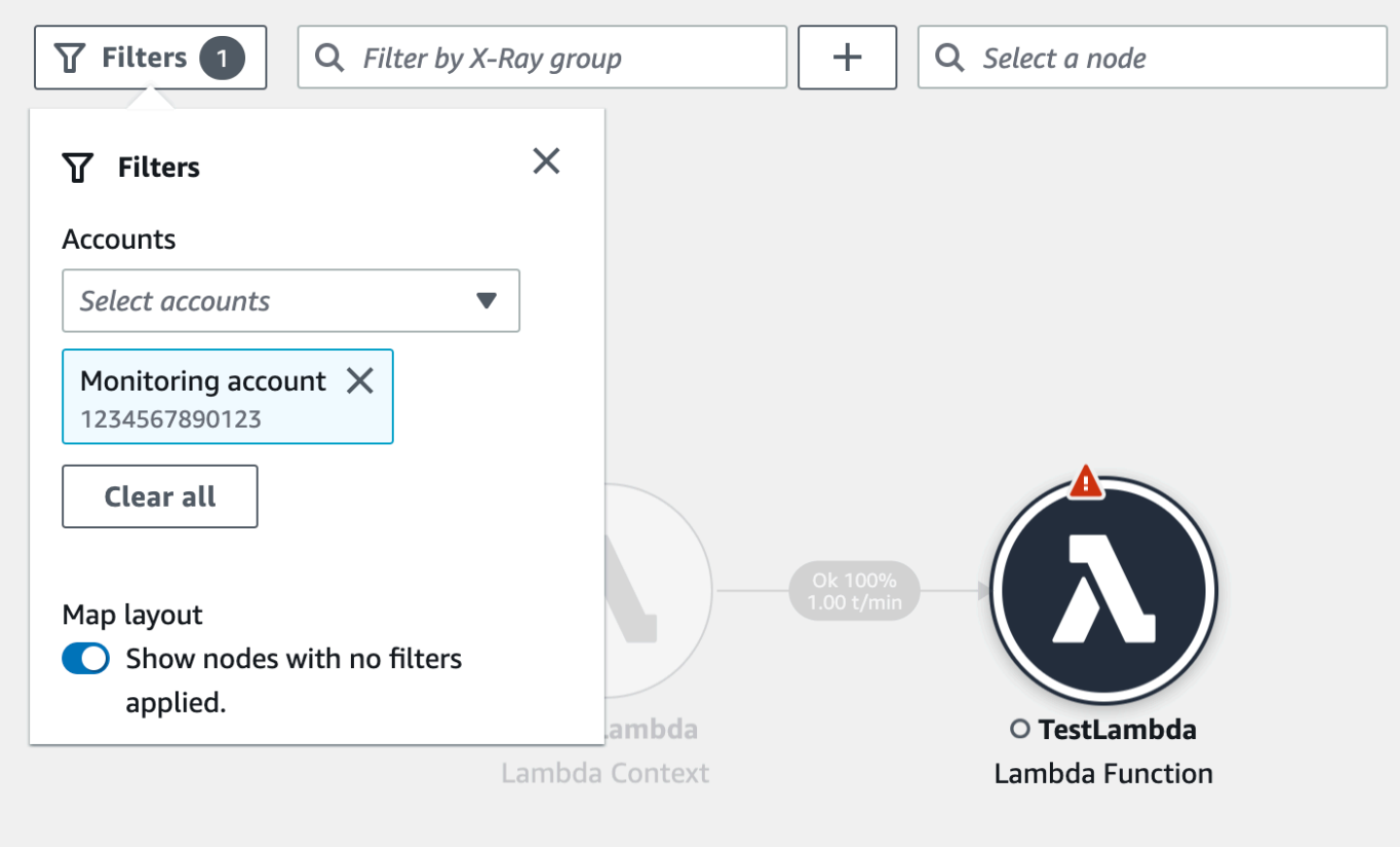
Kontoübergreifende Ablaufverfolgungen werden im Überwachungskonto angezeigt. Für jedes Quellkonto werden nur lokale Ablaufverfolgungen für dieses spezifische Konto angezeigt. In den

folgenden Abschnitten wird davon ausgegangen, dass Sie beim Monitoring-Konto angemeldet sind und die CloudWatch Amazon-Konsole geöffnet haben. Sowohl auf der Trace-Map- als auch auf der Traces-Seite wird in der oberen rechten Ecke ein Abzeichen für das Überwachungskonto angezeigt.

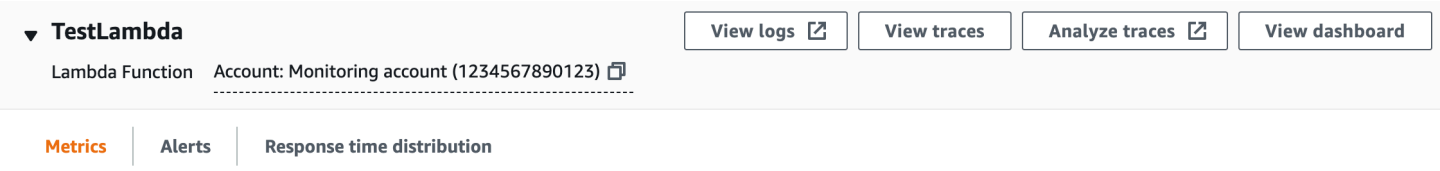


Karte verfolgen

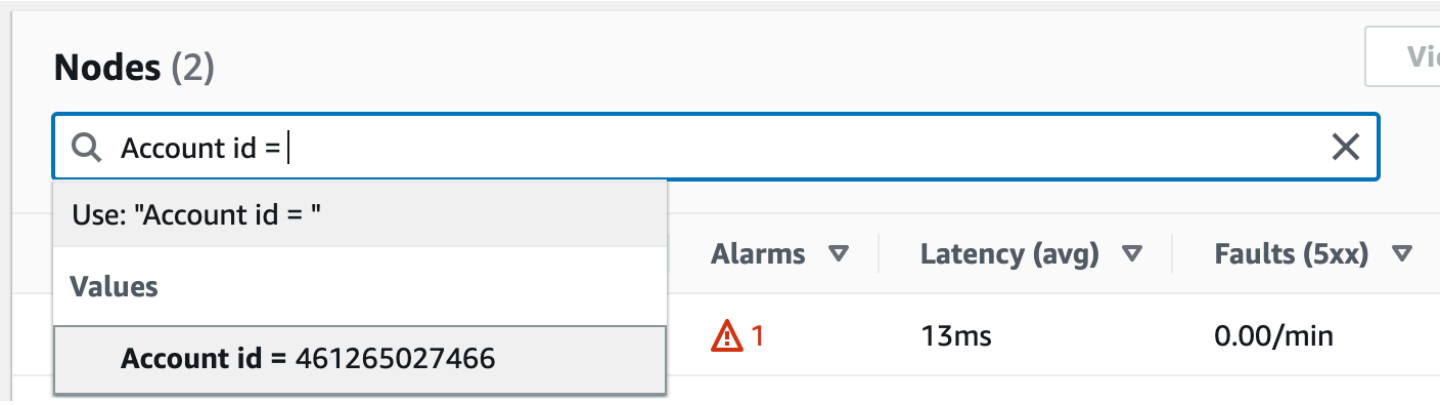
Wählen Sie in der CloudWatch Konsole im linken Navigationsbereich unter X-Ray-Traces die Option Trace Map aus. In der Trace-Map werden standardmäßig Knoten für alle Quellkonten angezeigt, die Traces an das Überwachungskonto senden, sowie Knoten für das Überwachungskonto selbst. Wählen Sie auf der Trace-Map oben links Filter aus, um die Trace-Map mithilfe der Drop-down-Liste Konten zu filtern. Nachdem ein Kontofilter angewendet wurde, sind Dienstknoten von Konten, die nicht dem aktuellen Filter entsprechen, ausgegraut.



Wenn Sie einen Dienstknoten auswählen, enthält der Bereich mit den Knotendetails die Konto-ID und das Label des Dienstes.



Wählen Sie in der oberen rechten Ecke der Trace-Map die Option Listenansicht aus, um eine Liste der Dienstknoten anzuzeigen. Die Liste der Dienstknoten umfasst Dienste aus dem Überwachungskonto und allen konfigurierten Quellkonten. Filtern Sie die Liste der Knoten nach Kontoname oder Konto-ID, indem Sie sie aus dem Knotenfilter auswählen.

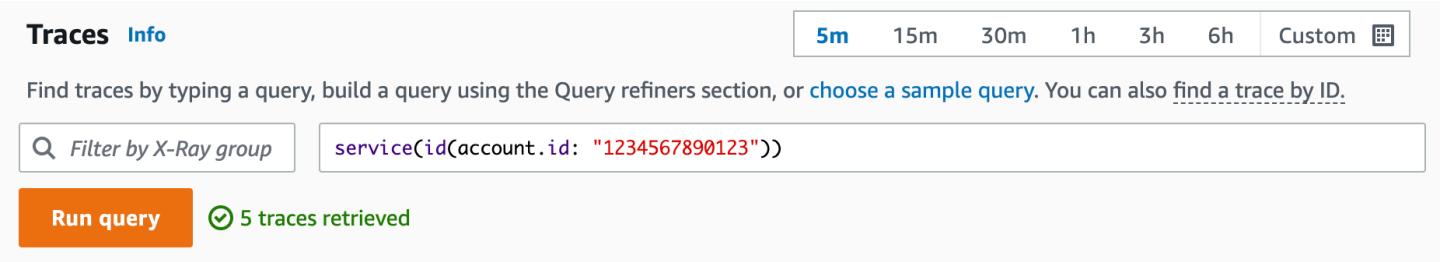


Ablaufverfolgungen

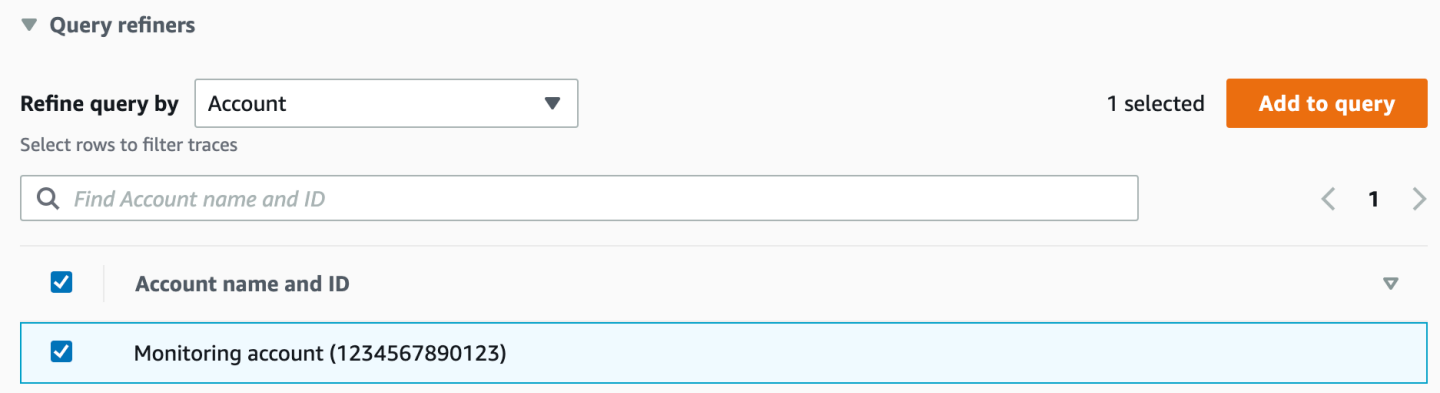
Rufen Sie die Trace-Details für Traces auf, die sich über mehrere Konten erstrecken, indem Sie die CloudWatch Konsole vom Monitoring-Konto aus öffnen und im linken Navigationsbereich unter X-Ray-Traces die Option Traces auswählen. Sie können diese Seite auch öffnen, indem Sie einen Knoten in der X-Ray-Trace-Map auswählen und dann im Bereich mit den Knotendetails die Option Traces anzeigen wählen.

Die Seite „Traces“ unterstützt Abfragen nach Konto-ID. Geben Sie zunächst eine Abfrage ein, die ein oder mehrere Konten IDs umfasst. Im folgenden Beispiel werden Traces abgefragt, die die Konto-ID X oder Y durchlaufen haben:

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



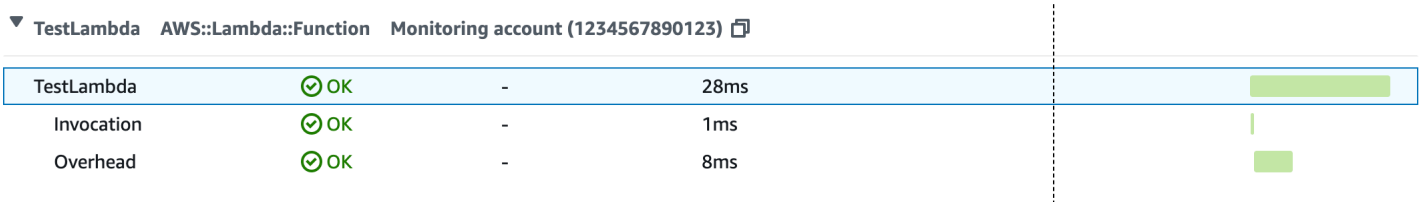
Verfeinern Sie Ihre Anfrage nach Konto. Wählen Sie ein oder mehrere Konten aus der Liste aus und klicken Sie dann auf Zur Abfrage hinzufügen.



Einzelheiten verfolgen

Sehen Sie sich die Details einer Ablaufverfolgung an, indem Sie sie aus der Traces-Liste unten auf der Traces-Seite auswählen. Die Trace-Details werden angezeigt, einschließlich einer Übersicht mit Ablaufverfolgungsdetails mit Dienstknoten aus allen Konten, die der Trace durchlaufen hat. Wählen Sie einen bestimmten Dienstknoten aus, um das entsprechende Konto anzuzeigen.

Im Abschnitt Segment-Timeline werden die Kontodetails für jedes Segment in der Timeline angezeigt.



Nachverfolgung ereignisgesteuerter Anwendungen

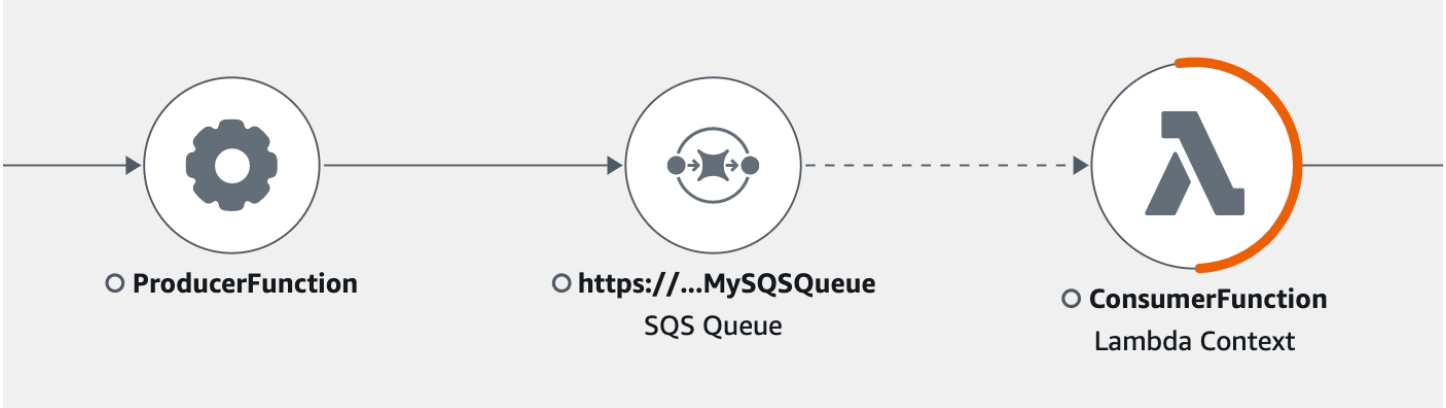
AWS X-Ray unterstützt die Ablaufverfolgung ereignisgesteuerter Anwendungen mithilfe von Amazon SQS und. AWS Lambda Verwenden Sie die CloudWatch Konsole, um eine verbundene Ansicht jeder Anfrage zu sehen, während sie bei Amazon SQS in die Warteschlange gestellt und von einer oder mehreren Lambda-Funktionen verarbeitet wird. Traces von Upstream-Nachrichtenproduzenten werden automatisch mit Traces von nachgeschalteten Lambda-Consumer-Knoten verknüpft, wodurch eine end-to-end Ansicht der Anwendung erstellt wird.

> **Note**
>
> Jedes Trace-Segment kann mit bis zu 20 Traces verknüpft werden, wobei ein Trace maximal 100 Links enthalten kann. In bestimmten Szenarien kann das Verknüpfen zusätzlicher

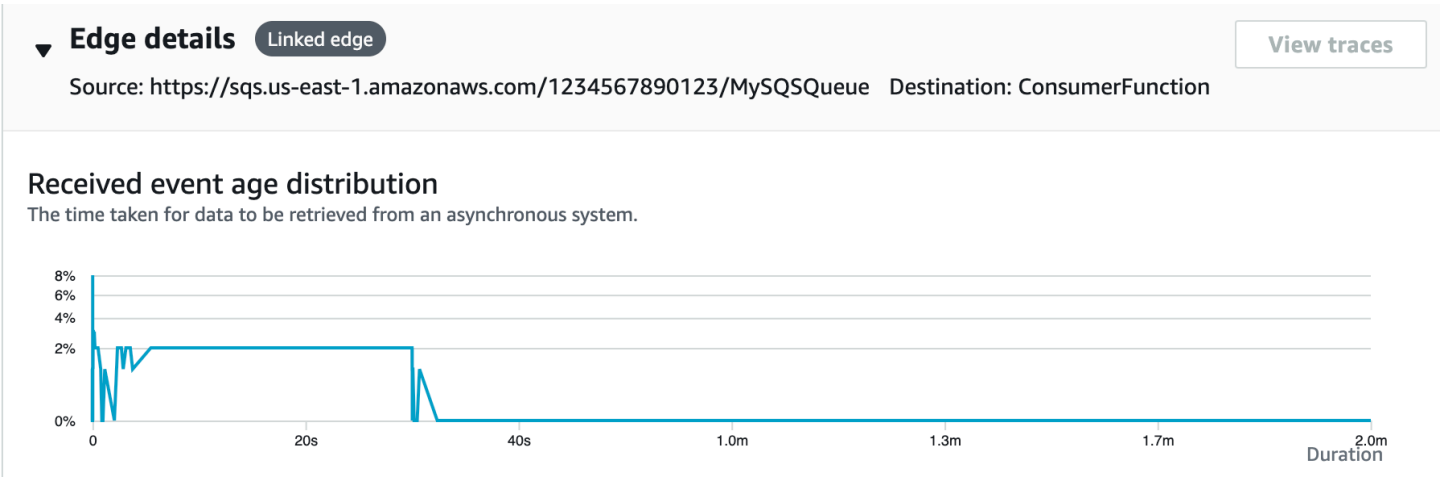
Spuren dazu führen, dass die maximale Größe des Trace-Dokuments überschritten wird, was zu einer möglicherweise unvollständigen Ablaufverfolgung führen kann. Dies kann beispielsweise passieren, wenn eine Lambda-Funktion mit aktivierter Ablaufverfolgung viele SQS-Nachrichten in einem einzigen Aufruf an eine Warteschlange sendet. Wenn Sie auf dieses Problem stoßen, ist eine Abhilfemaßnahme verfügbar, die X-Ray SDKs verwendet. Weitere Informationen finden Sie im X-Ray-SDK für Java, Node.js, Python, Go oder .NET.

Verknüpfte Traces in der Trace-Map anzeigen

Verwenden Sie die Trace-Map-Seite in der CloudWatchKonsole, um eine Trace-Map mit Traces von Nachrichtenproduzenten anzuzeigen, die mit Traces von Lambda-Verbrauchern verknüpft sind. Diese Links werden mit einer gestrichelten Linie angezeigt, die den Amazon SQS-Knoten mit den nachgeschalteten Lambda-Consumer-Knoten verbindet.



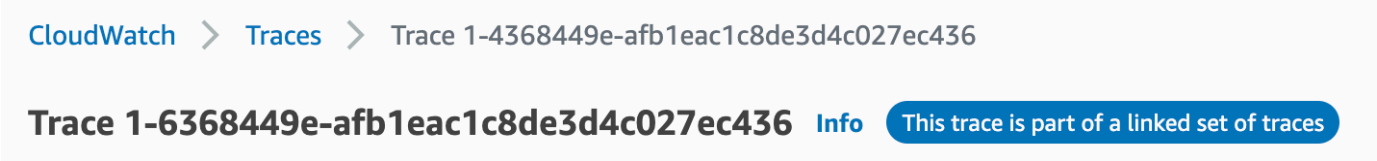
Wählen Sie eine gestrichelte Linie aus, um ein Histogramm mit dem Alter eines empfangenen Ereignisses anzuzeigen, das die Verteilung des Ereignisalters bei Empfang durch Verbraucher darstellt. Das Alter wird jedes Mal berechnet, wenn ein Ereignis empfangen wird.



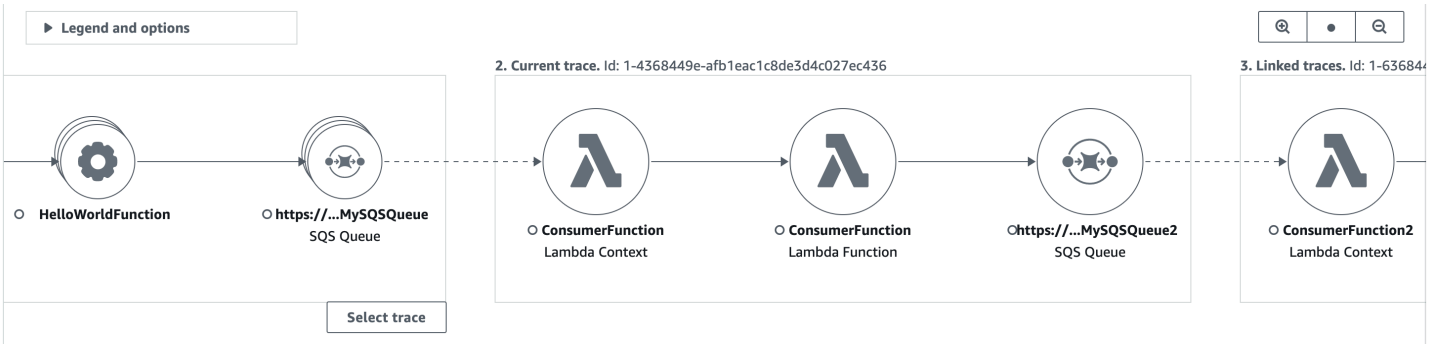
Details der verknüpften Ablaufverfolgung anzeigen

Trace-Details anzeigen, die von einem Nachrichtenproduzenten, einer Amazon SQS SQS-Warteschlange oder einem Lambda-Consumer gesendet wurden:

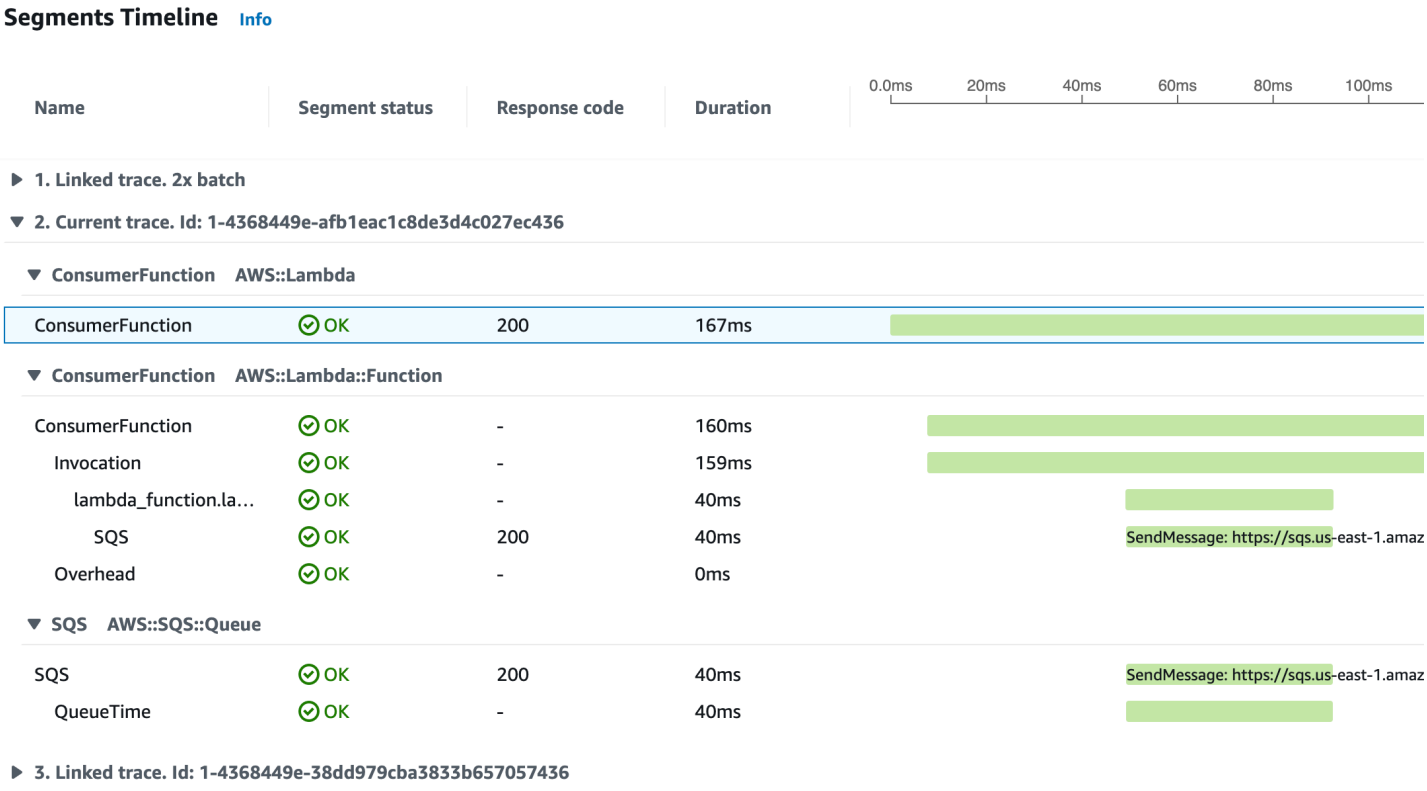
1. Verwenden Sie die Trace-Map, um einen Message Producer, Amazon SQS oder Lambda Consumer Node auszuwählen.
2. Wählen Sie im Bereich mit den Knotendetails die Option Traces anzeigen, um eine Liste von Traces anzuzeigen. Sie können in der CloudWatch Konsole auch direkt zur Seite „Traces“ navigieren.
3. Wählen Sie einen bestimmten Trace aus der Liste aus, um die Seite mit den Trace-Details zu öffnen. Auf der Seite mit den Ablaufverfolgungsdetails wird eine Meldung angezeigt, wenn der ausgewählte Trace Teil eines verknüpften Trace-Satzes ist.



In der Karte mit den Verfolgungsdetails werden die aktuelle Spur zusammen mit den verknüpften Spuren flussaufwärts und flussabwärts angezeigt, die jeweils in einem Feld enthalten sind, das die Grenzen der einzelnen Spuren angibt. Wenn die aktuell ausgewählte Spur mit mehreren Upstream- oder Downstream-Leiterbahnen verknüpft ist, werden die Knoten innerhalb der Upstream- oder Downstream-verknüpften Spuren gestapelt, und die Schaltfläche „Spur auswählen“ wird angezeigt.



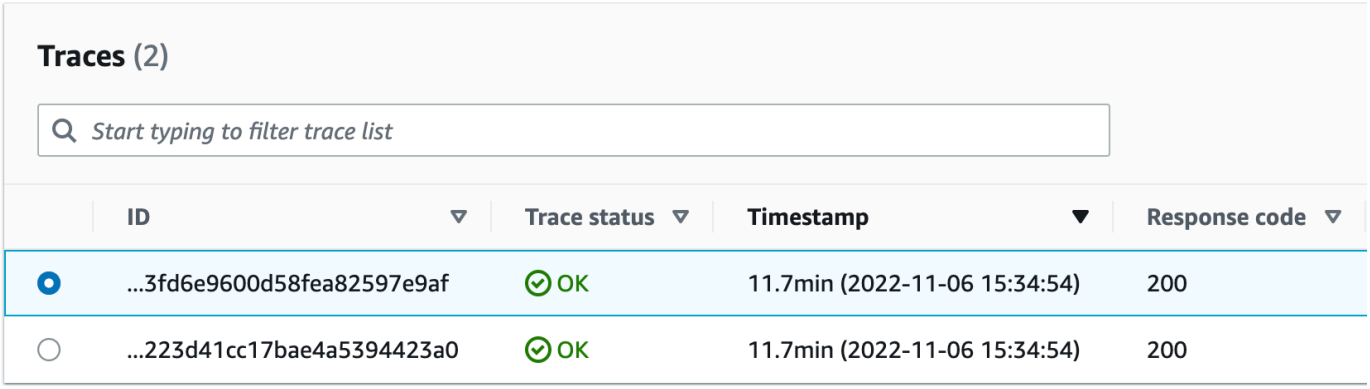
Unter der Karte mit den Spurdetails wird eine Zeitleiste mit Spurensegmenten angezeigt, einschließlich verknüpfter Spuren flussaufwärts und flussabwärts. Wenn mehrere stromaufwärts oder flussabwärts verknüpfte Spuren vorhanden sind, können deren Segmentdetails nicht angezeigt werden. Um Segmentdetails für eine einzelne Spur innerhalb einer Reihe verknüpfter Spuren anzuzeigen, wählen Sie eine einzelne Spur aus, wie unten beschrieben.



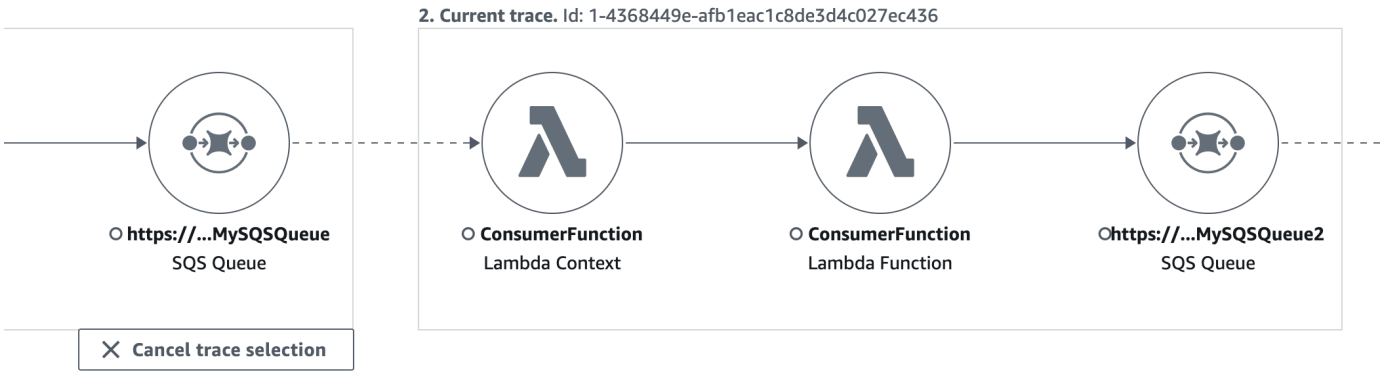
Wählen Sie eine einzelne Spur innerhalb einer Reihe verknüpfter Spuren aus

Filtert einen verknüpften Satz von Spuren zu einer einzelnen Spur, um Segmentdetails in der Timeline zu sehen.

1. Wählen Sie auf der Karte mit den Trace-Details unter den verknüpften Spuren die Option Spur auswählen aus. Eine Liste von Spuren wird angezeigt.



2. Wählen Sie das Optionsfeld neben einer Spur aus, um sie in der Karte mit den Ablaufverfolgungsdetails anzuzeigen.
3. Wählen Sie „Spurauswahl abbrechen“, um den gesamten Satz verknüpfter Spuren anzuzeigen.



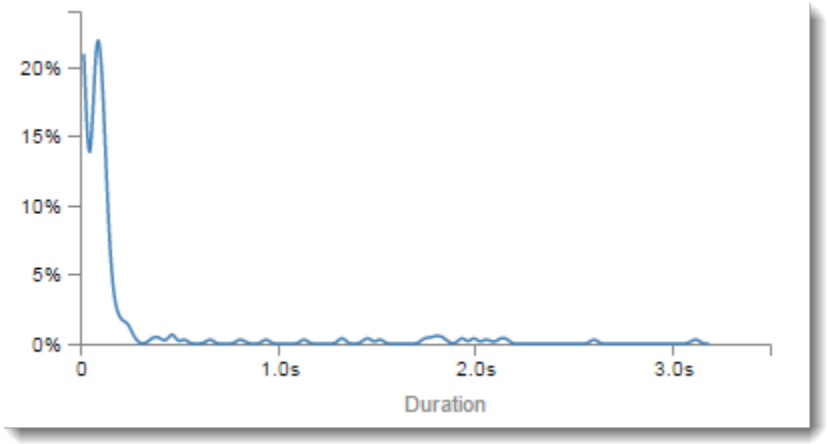
Verwendung von Latenzhistogrammen

Wenn Sie einen Knoten oder eine Kante auf einer AWS X-Ray Trace-Map auswählen, zeigt die X-Ray-Konsole ein Latenzverteilungshistogramm an.

Latency

Als Latenz wird die Zeit zwischen dem Start und dem Abschluss einer Anforderung bezeichnet. Ein Histogramm zeigt eine Verteilung von Latenzen. Er zeigt Dauer auf der x-Achse und den Prozentsatz der Anforderungen, die jeder Dauer entsprechen, auf der y-Achse.

Dieses Histogramm zeigt einen Service, der die meisten Anfragen in weniger als 300 ms abschließt. Ein kleiner Prozentsatz von Anfragen dauert 2 Sekunden, und einige Ausreißer nehmen noch mehr Zeit in Anspruch.



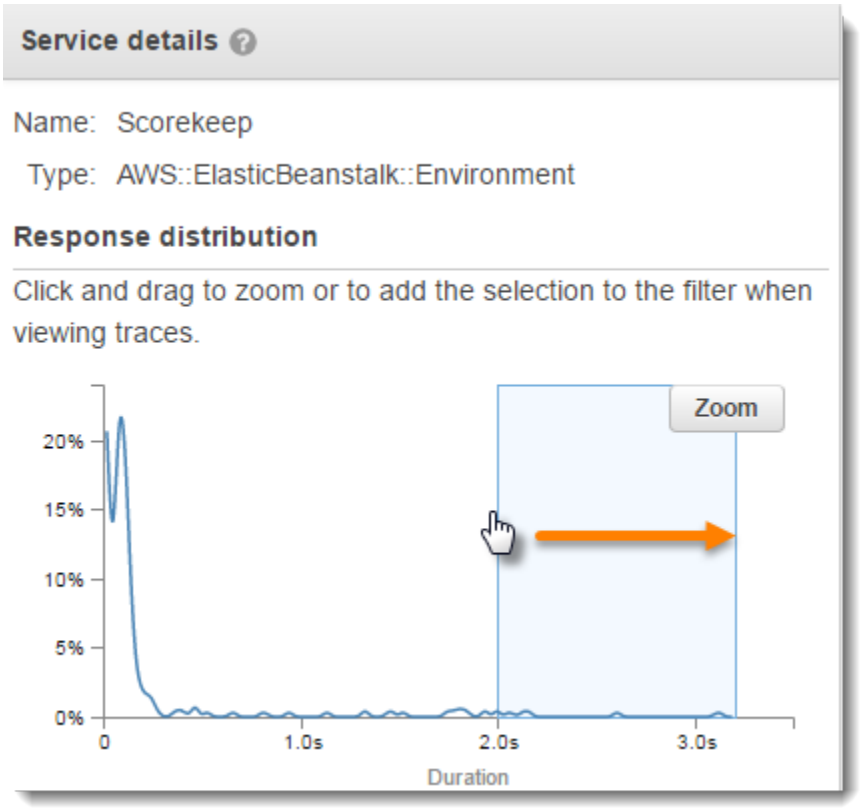
Interpretieren von Service-Details

Service-Histogramme und Edge-Histogramme bieten eine visuelle Darstellung der Latenz aus der Sicht eines Services oder eines Anforderers.

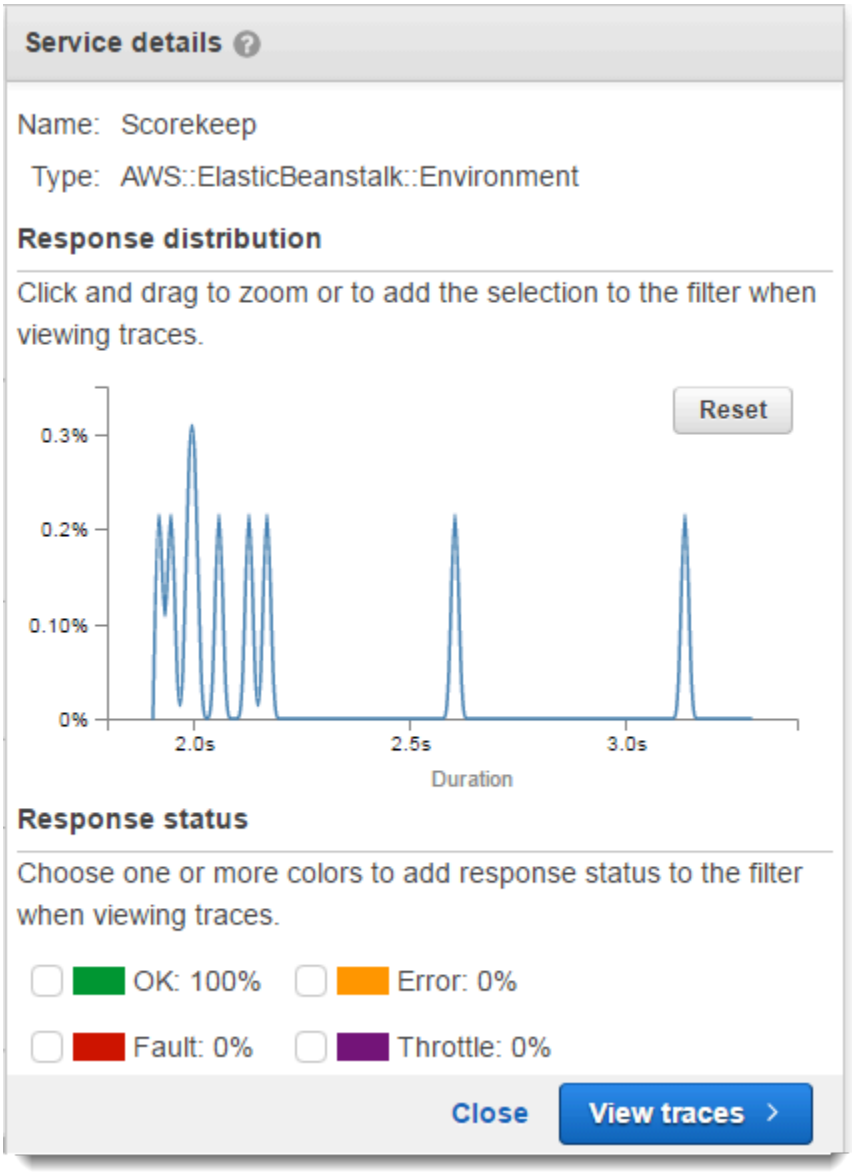
- Wählen Sie einen Dienstknoten aus, indem Sie auf den Kreis klicken. X-Ray zeigt ein Histogramm für Anfragen, die vom Dienst bedient werden. Die Latenzen sind die, die der Service aufgezeichnet hat; dazu gehört nicht die Netzwerklatenz zwischen dem Service und dem Auftraggeber.
- Wählen Sie eine Kante aus, indem Sie auf die Linie oder die Pfeilspitze der Kante zwischen zwei Diensten klicken. X-Ray zeigt ein Histogramm für Anfragen des Anforderers, die vom Downstream-Dienst bearbeitet wurden. Die Latenzen sind die vom Auftraggeber aufgezeichneten; dazu gehört die Latenz in der Netzwerkverbindung zwischen den beiden Services.

Zur Interpretation des Panel-Histogramms mit den Servicedetails können Sie die Werte betrachten, die am stärksten von den meisten Werten im Histogramm abweichen. Diese Ausreißer sind als Spitzen im Histogramm zu sehen und Sie können die Ablaufverfolgungen für einen bestimmten Bereich untersuchen, um zu sehen, was vor sich geht.

Wählen Sie zur Anzeige nach Latenz gefilterter Ablaufverfolgungen einen Bereich im Histogramm aus. Klicken Sie an die Stelle, an der die Auswahl beginnen soll, und ziehen Sie von links nach rechts, um einen Bereich von Latenzen hervorzuheben, der in den Ablaufverfolgungsfilter eingeschlossen werden soll.



Nach Auswahl eines Bereichs können Sie den Zoom wählen, um nur diesen Teil des Histogramms anzuzeigen, und Ihre Auswahl verfeinern.



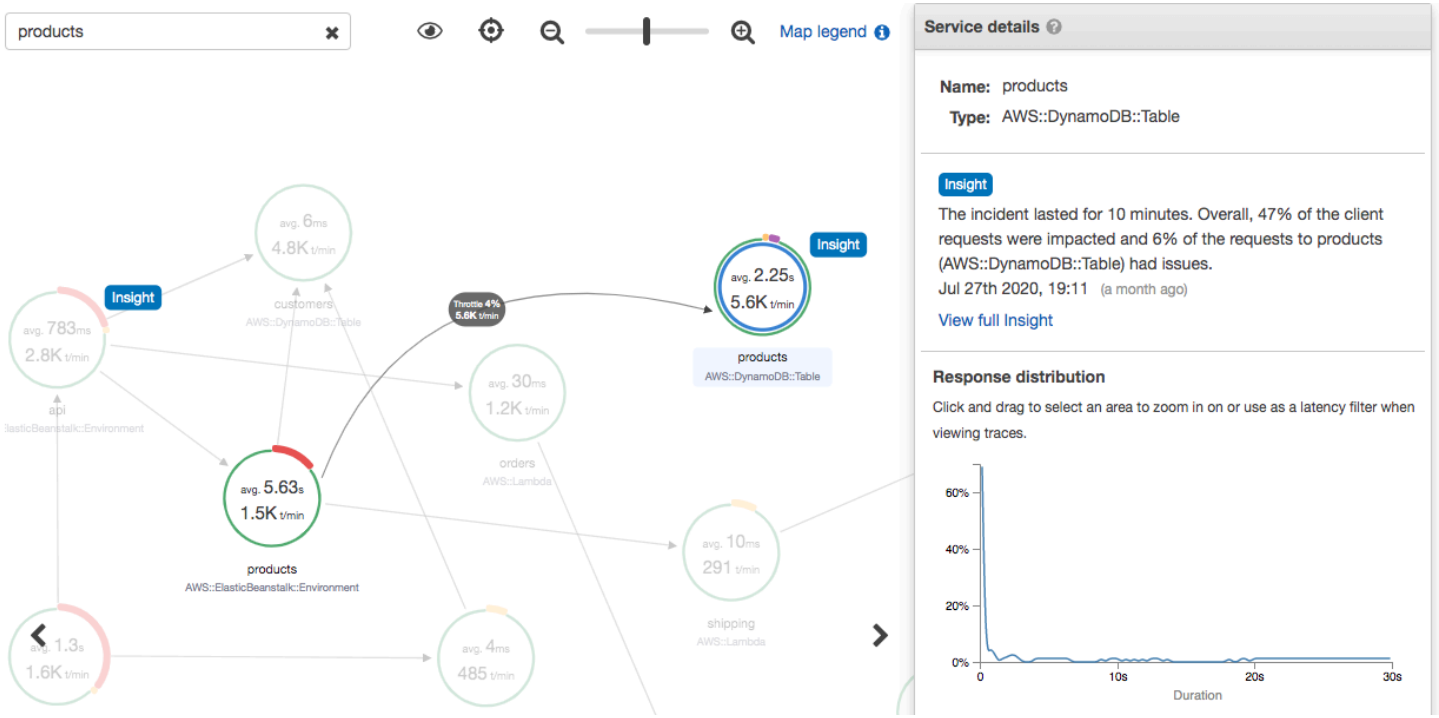
Sobald Sie den Fokus auf den interessierenden Bereich gesetzt haben, wählen Sie Ablaufverfolgungen anzeigen.

Nutzung von X-Ray Insights

AWS X-Ray analysiert kontinuierlich die Trace-Daten in Ihrem Konto, um neu auftretende Probleme in Ihren Anwendungen zu identifizieren. Wenn die Fehlerraten den erwarteten Bereich überschreiten, werden Erkenntnisse gewonnen, die das Problem aufzeichnen und seine Auswirkungen verfolgen, bis es behoben ist. Mithilfe von Erkenntnissen können Sie:

- Identifizieren Sie, wo in Ihrer Anwendung Probleme auftreten, die Hauptursache des Problems und die damit verbundenen Auswirkungen. Anhand der von Insights bereitgestellten Auswirkungsanalyse können Sie den Schweregrad und die Priorität eines Problems ableiten.
- Erhalten Sie Benachrichtigungen, wenn sich das Problem im Laufe der Zeit ändert. Insights-Benachrichtigungen können mithilfe von Amazon EventBridge in Ihre Überwachungs- und Warnlösung integriert werden. Diese Integration ermöglicht es Ihnen, je nach Schweregrad des Problems automatisierte E-Mails oder Benachrichtigungen zu versenden.

Die X-Ray-Konsole identifiziert Knoten mit laufenden Vorfällen in der Trace-Map. Um eine Zusammenfassung der Erkenntnisse zu erhalten, wählen Sie den betroffenen Knoten aus. Sie können Einblicke auch anzeigen und filtern, indem Sie im Navigationsbereich auf der linken Seite Insights auswählen.



X-Ray generiert einen Einblick, wenn es eine Anomalie in einem oder mehreren Knoten der Service-Map erkennt. Der Service verwendet statistische Modelle, um die erwarteten Fehlerraten der Dienste in Ihrer Anwendung vorherzusagen. Im vorherigen Beispiel handelt es sich bei der Anomalie um eine Zunahme von Fehlern von AWS Elastic Beanstalk. Auf dem Elastic Beanstalk-Server kam es zu mehreren Timeouts bei API-Aufrufen, was zu einer Anomalie in den Downstream-Knoten führte.

Einblicke in der X-Ray-Konsole aktivieren

Insights muss für jede Gruppe aktiviert sein, mit der Sie Insights-Funktionen verwenden möchten. Sie können Insights auf der Gruppenseite aktivieren.

1. Öffnen Sie die X-Ray-Konsole.
2. Wählen Sie eine bestehende Gruppe aus oder erstellen Sie eine neue, indem Sie Gruppe erstellen und dann Insights aktivieren auswählen. Weitere Informationen zur Konfiguration von Gruppen in der X-Ray-Konsole finden Sie unterGruppen konfigurieren.
3. Wählen Sie im Navigationsbereich auf der linken Seite Insights und dann einen Insight aus, den Sie sich ansehen möchten.

> **Note**
>
> X-Ray verwendet GetInsightSummaries GetInsight, GetInsightEvents, und GetInsightImpactGraph API-Operationen, um Daten aus Insights abzurufen. Weitere Informationen finden Sie unter Wie AWS X-Ray funktioniert mit IAM.

Aktivieren Sie Insights-Benachrichtigungen

Bei Insights-Benachrichtigungen wird für jedes Insight-Ereignis eine Benachrichtigung erstellt, z. B. wenn ein Insight erstellt wird, sich erheblich ändert oder geschlossen wird. Kunden können diese Benachrichtigungen über EventBridge Amazon-Ereignisse erhalten und bedingte Regeln verwenden, um Aktionen wie SNS-Benachrichtigungen, Lambda-Aufrufe, das Posten von Nachrichten in eine SQS-Warteschlange oder eines der unterstützten Ziele zu ergreifen. EventBridge Insights-Benachrichtigungen werden nach bestem Wissen versendet, können aber nicht garantiert werden. Weitere Informationen zu Zielen finden Sie unter Amazon EventBridge Targets.

Auf der Gruppenseite können Sie Insights-Benachrichtigungen für jede Gruppe aktivieren, für die Insights aktiviert sind.

Um Benachrichtigungen für eine X-Ray-Gruppe zu aktivieren

1. Öffnen Sie die X-Ray-Konsole.
2. Wählen Sie eine bestehende Gruppe aus oder erstellen Sie eine neue, indem Sie Gruppe erstellen wählen. Vergewissern Sie sich, dass Insights aktivieren ausgewählt ist, und wählen Sie dann Benachrichtigungen aktivieren aus. Weitere Informationen zur Konfiguration von Gruppen in der X-Ray-Konsole finden Sie unterGruppen konfigurieren.

So konfigurieren Sie EventBridge bedingte Amazon-Regeln

1. Öffnen Sie die EventBridge Amazon-Konsole.
2. Navigieren Sie in der linken Navigationsleiste zu Regeln und wählen Sie Regel erstellen aus.
3. Geben Sie einen Namen und eine Beschreibung für die Regel ein.
4. Wählen Sie Ereignismuster und anschließend Benutzerdefiniertes Muster aus. Geben Sie ein Muster an, das `"source": [ "aws.xray" ]` und enthält`"detail-type": [ "AWS X-Ray Insight Update" ]`. Im Folgenden finden Sie einige Beispiele für mögliche Muster.
   - Ereignismuster, das allen eingehenden Ereignissen aus X-Ray Insights entspricht:

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

   - Ereignismuster, das einem bestimmten Wert entspricht **`state`** und**`category`**:

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

5. Wählen und konfigurieren Sie die Ziele, die Sie aufrufen möchten, wenn ein Ereignis dieser Regel entspricht.
6. (Optional) Geben Sie Tags an, um diese Regel leichter identifizieren und auswählen zu können.

7. Wählen Sie Erstellen aus.

> **Note**
>
> X-Ray Insights-Benachrichtigungen senden Ereignisse an Amazon EventBridge, das derzeit keine vom Kunden verwalteten Schlüssel unterstützt. Weitere Informationen finden Sie unter Datenschutz in AWS X-Ray.

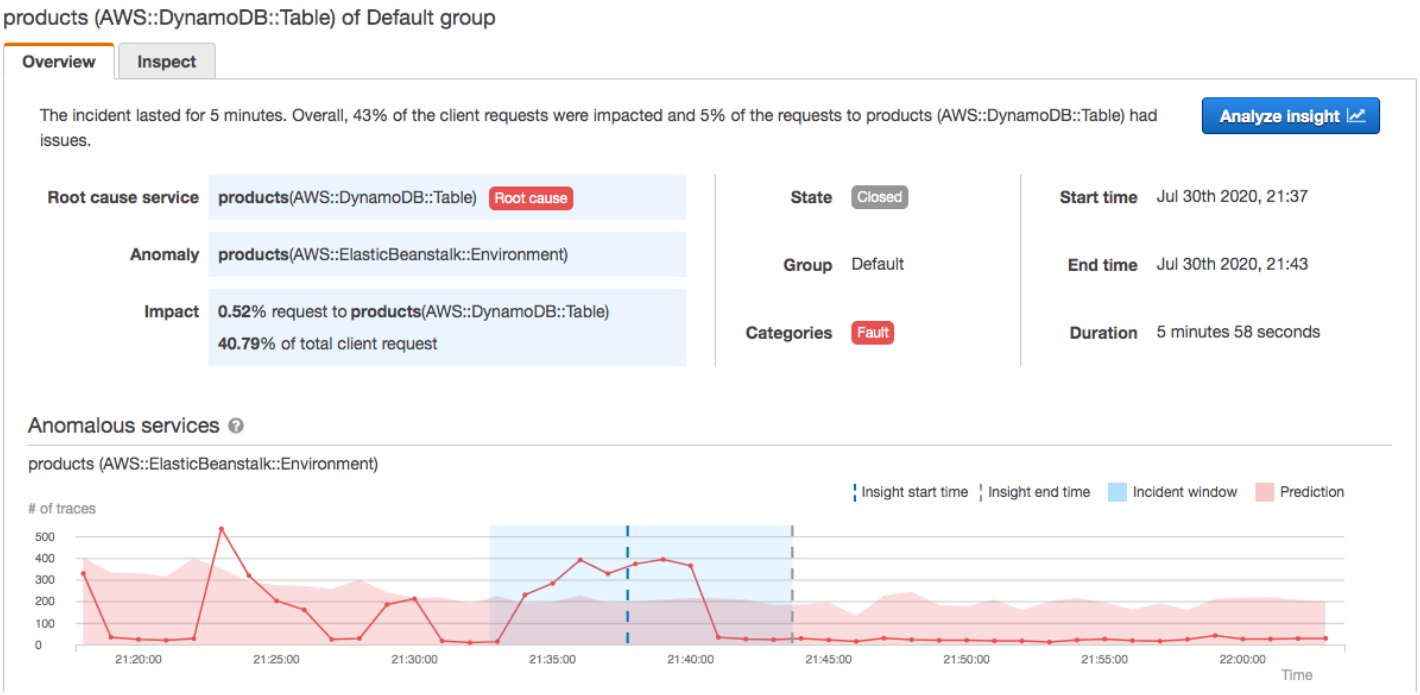
Überblick über Insight

Die Übersichtsseite für einen Einblick versucht, drei wichtige Fragen zu beantworten:

- Was ist das zugrundeliegende Problem?
- Was ist die Hauptursache?
- Was ist die Auswirkung?

Der Abschnitt Anomale Dienste zeigt für jeden Service einen Zeitplan, der die Veränderung der Fehlerraten während des Vorfalls veranschaulicht. Die Zeitleiste zeigt die Anzahl der Traces mit Fehlern auf einem durchgehenden Band, das die erwartete Anzahl von Fehlern auf der Grundlage des aufgezeichneten Datenverkehrs angibt. Die Dauer der Erkenntnisse wird im Incident-Fenster visualisiert. Das Incident-Fenster beginnt, wenn X-Ray beobachtet, dass die Metrik anomal wird, und bleibt bestehen, solange der Insight aktiv ist.

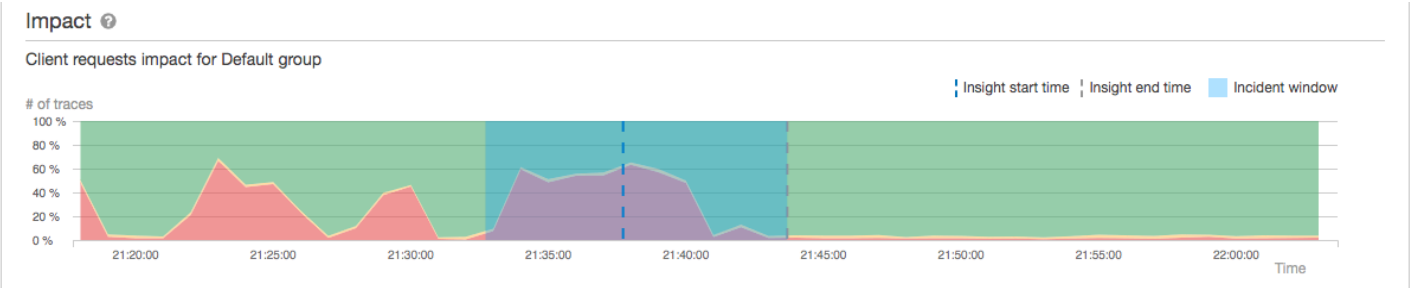
Das folgende Beispiel zeigt eine Zunahme von Fehlern, die zu einem Vorfall geführt haben:



Im Abschnitt „Ursache“ wird eine Ablaufverfolgungsübersicht angezeigt, die sich auf den Ursachendienst und den betroffenen Pfad konzentriert. Sie können die nicht betroffenen Knoten ausblenden, indem Sie auf das Augensymbol oben rechts in der Ursachenkarte klicken. Der Root Cause Service ist der am weitesten flussabwärts gelegene Knoten, an dem X-Ray eine Anomalie identifiziert hat. Dabei kann es sich um einen Service handeln, den Sie instrumentiert haben, oder um einen externen Service, den Ihr Service mit einem instrumentierten Client aufgerufen hat. Wenn Sie beispielsweise Amazon DynamoDB mit einem instrumentierten AWS SDK-Client aufrufen, führt eine Zunahme von Fehlern durch DynamoDB zu Erkenntnissen mit DynamoDB als Hauptursache.

Um die Ursache weiter zu untersuchen, wählen Sie im Ursachendiagramm die Option Ursachendetails anzeigen aus. Sie können die Analytics-Seite verwenden, um die Grundursache und zugehörige Meldungen zu untersuchen. Weitere Informationen finden Sie unter Interaktion mit der -Analysekonsole.

Fehler, die sich in der Map weiter flussaufwärts fortsetzen, können sich auf mehrere Knoten auswirken und mehrere Anomalien verursachen. Wenn ein Fehler vollständig an den Benutzer weitergegeben wird, der die Anfrage gestellt hat, ist das Ergebnis ein Client-Fehler. Dies ist ein Fehler im Stammknoten der Trace-Map. Das Impact-Diagramm bietet einen Überblick über das Kundenerlebnis für die gesamte Gruppe. Dieses Kundenerlebnis wird anhand der Prozentsätze der folgenden Zustände berechnet: Fehler, Fehler, Drosselung und Okay.



Dieses Beispiel zeigt eine Zunahme von Traces mit einem Fehler am Stammknoten während eines Vorfalls. Vorfälle in nachgelagerten Diensten entsprechen nicht immer einer Zunahme von Client-Fehlern.

Wenn Sie Analyze Insight wählen, wird die X-Ray Analytics-Konsole in einem Fenster geöffnet, in dem Sie tief in die Spuren eintauchen können, die den Einblick verursacht haben. Weitere Informationen finden Sie unter Interaktion mit der -Analysekonsole.

Auswirkungen verstehen

AWS X-Ray misst im Rahmen der Generierung von Erkenntnissen und Benachrichtigungen die Auswirkungen, die durch ein laufendes Problem verursacht werden. Die Auswirkungen werden auf zwei Arten gemessen:

- Auswirkungen auf die X-Ray-Gruppe
- Auswirkungen auf den Ursachendienst

Diese Auswirkung wird durch den Prozentsatz der Anfragen bestimmt, die innerhalb eines bestimmten Zeitraums fehlschlagen oder einen Fehler verursachen. Diese Auswirkungsanalyse ermöglicht es Ihnen, den Schweregrad und die Priorität des Problems anhand Ihres speziellen Szenarios abzuleiten. Diese Auswirkung ist zusätzlich zu den Insights-Benachrichtigungen als Teil des Konsolen-Erlebnisses verfügbar.

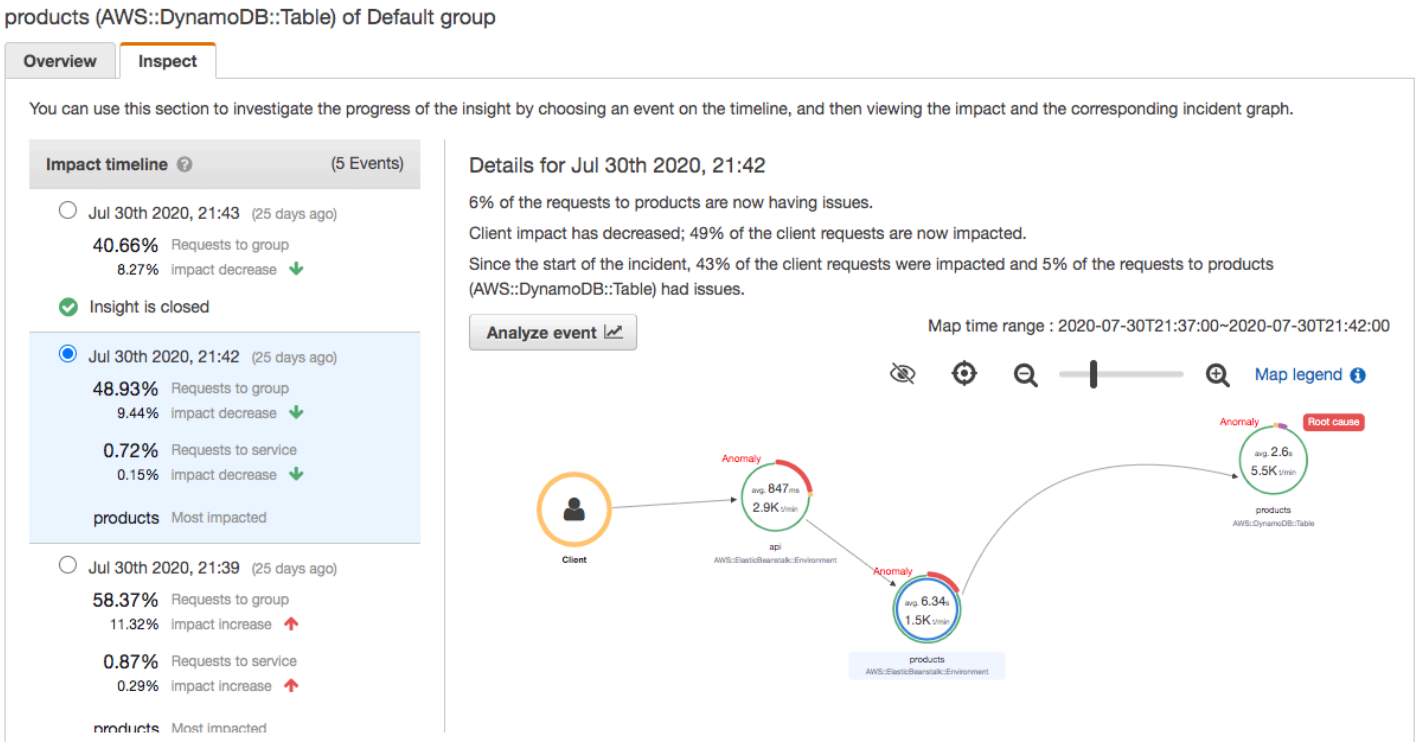
Deduplizierung

AWS X-Ray Insights dedupliziert Probleme bei mehreren Microservices. Es verwendet die Anomalieerkennung, um den Service zu ermitteln, der die Hauptursache eines Problems ist, ermittelt, ob andere verwandte Dienste aufgrund derselben Grundursache ein anomales Verhalten zeigen, und zeichnet das Ergebnis in Form eines einzigen Einblicks auf.

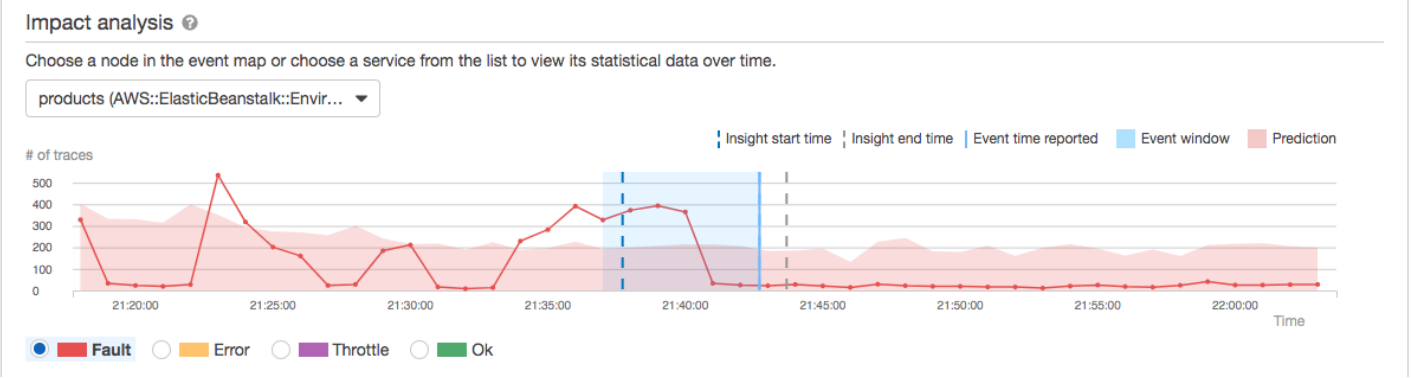
Überprüfen Sie den Fortschritt eines Insights

X-Ray bewertet die Erkenntnisse in regelmäßigen Abständen neu, bis sie behoben sind, und zeichnet jede wichtige zwischenzeitliche Änderung als Benachrichtigung auf, die als EventBridge Amazon-Ereignis gesendet werden kann. Auf diese Weise können Sie Prozesse und Workflows erstellen, um festzustellen, wie sich das Problem im Laufe der Zeit verändert hat, und geeignete Maßnahmen ergreifen, z. B. das Senden einer E-Mail oder die Integration in ein Warnsystem mithilfe von. EventBridge

Sie können Vorfallereignisse in der Impact-Timeline auf der Seite Inspect überprüfen. Standardmäßig zeigt die Zeitleiste den Service an, der am stärksten betroffen ist, bis Sie einen anderen Service auswählen.



Um eine Ablaufkarte und Grafiken für ein Ereignis zu sehen, wählen Sie es aus der Wirkungszeitleiste aus. Die Trace-Map zeigt die Dienste in Ihrer Anwendung, die von dem Vorfall betroffen sind. Unter Auswirkungsanalyse zeigen Diagramme die Fehlerzeitpläne für den ausgewählten Knoten und für die Clients in der Gruppe.



Wenn Sie sich die Spuren eines Vorfalls genauer ansehen möchten, wählen Sie auf der Seite Inspizieren die Option Ereignis analysieren aus. Sie können die Analytics-Seite verwenden, um die Liste der Traces zu verfeinern und die betroffenen Benutzer zu identifizieren. Weitere Informationen finden Sie unter Interaktion mit der -Analysekonsole.

Interaktion mit der -Analysekonsole

Die AWS X-Ray Analytics-Konsole ist ein interaktives Tool zur Interpretation von Trace-Daten, mit dem Sie schnell nachvollziehen können, wie Ihre Anwendung und die zugrunde liegenden Dienste funktionieren. Die Konsole ermöglicht Ihnen das Erkunden, Analysieren und Visualisieren von Ablaufverfolgungen durch interaktive Reaktionszeit und Zeitreihen-Diagramme.

Bei der Auswahl in der Analysekonsole erstellt die Konsole Filter für die ausgewählte Teilmenge aller Ablaufverfolgungen. Sie können die aktiven Datasets mit zunehmend granularen Filtern anpassen, indem Sie auf die Diagramme und die Bereiche der Metriken und Felder klicken, die der aktuellen Ablaufverfolgungsreihe zugeordnet sind.

Themen

Konsolenfunktionen

Die X-Ray Analytics-Konsole verwendet die folgenden Hauptfunktionen zum Gruppieren, Filtern, Vergleichen und Quantifizieren von Trace-Daten.

Features

Funktion	Beschreibung
Gruppen	Die anfangs ausgewählte Gruppe ist `Default`. Zum Ändern der abgerufenen Gruppe wählen Sie eine andere Gruppe aus dem Menü rechts

Funktion	Beschreibung
	neben der Filterausdruck-Suchleiste aus. Weitere Informationen zu Gruppen finden Sie unter Verwenden von Filterausdrücken mit Gruppen.
Abgerufene Ablaufverfolgungen	Standardmäßig generiert die Analysekonsole Diagramme auf Grundlage aller Ablaufver folgungen in der ausgewählten Gruppe. Abgerufene Ablaufverfolgungen stehen für alle Ablaufverfolgungen in Ihrem Arbeitssatz. Auf dieser Kachel finden Sie die Anzahl der Ablaufverfolgungen. Mit Filterausdrücken an der Hauptsuchleiste optimieren und aktualisi eren Sie die abgerufenen Ablaufverfolgungen.
In Diagrammen anzeigen/In Diagrammen ausblenden	Ein Schalter zum Vergleichen der aktiven Gruppe mit den abgerufenen Ablaufver folgungen. Zum Vergleichen der Daten im Zusammenhang mit der Gruppe mit sämtliche n aktiven Filtern klicken Sie auf In Diagramme n anzeigen. Zum Entfernen dieser Ansicht aus den Diagrammen wählen Sie In Diagrammen ausblenden.
Gefilterter Ablaufverfolgungssatz A	Wenden Sie durch Interaktionen mit den Grafiken und Tabellen Filter an, um die Kriterien für den gefilterten Trace-Satz A zu erstellen. Wenn die Filter angewendet werden, werden die Anzahl der zutreffenden Spuren und der Prozentsatz der abgerufenen Spuren an der Gesamtzahl der abgerufenen Spuren innerhalb dieser Kachel berechnet. Filter werden auf der Kachel Gefilterter Ablaufver folgungssatz A als Tags angezeigt und können zudem von der Kachel entfernt werden.

Funktion	Beschreibung
Optimieren	Über diese Funktion wird der Satz der abgerufenen Ablaufverfolgungen basierend auf den Filtern, die auf den Ablaufverfolgungss atz A angewendet wurden, aktualisiert. Durch Optimieren des abgerufenen Ablaufver folgungssatzes wird der Arbeitssatz aller abgerufenen Ablaufverfolgungen basierend auf den Filtern für den Ablaufverfolgungssatz A abgerufen. Der Arbeitssatz der abgerufenen Ablaufverfolgungen ist eine geprüfte Teilmenge aller Ablaufverfolgungen in der Gruppe.
Gefilterter Ablaufverfolgungssatz B	Bei der Erstellung ist der gefilterte Ablaufver folgungssatz B eine Kopie des gefilterten Ablaufverfolgungssatzes A. Um die beiden Verfolgungssätze zu vergleichen, wählen Sie einen neuen Filter aus, der für den Verfolgun gssatz B gilt, während der Verfolgungssatz A unverändert bleibt. Während der Anwendung der Filter werden die Anzahl der entsprech enden Ablaufverfolgungen und der Prozentsa tz der Ablaufverfolgungen von der Gesamtzahl der abgerufenen Ablaufverfolgungen auf dieser Kachel berechnet. Filter werden auf der Kachel Ablaufverfolgungssatz B als Tags angezeigt und können zudem von der Kachel entfernt werden.

Funktion	Beschreibung
Reaktionszeit Ursachenentitätspfade	Eine Tabelle mit aufgezeichneten Entitätsp faden. X-Ray bestimmt, welcher Pfad in Ihrer Spur die wahrscheinlichste Ursache für die Reaktionszeit ist. Das Format steht für eine Hierarchie von vorgefundenen Entitäten. Dies führt zu einer Reaktionszeit-Ursache. Verwenden Sie diese Zeilen, um nach wiederkehrenden Reaktionszeitfehle rn zu filtern. Weitere Informationen über das Anpassen eines Ursachenfilters und Schleusen von Daten durch die API finden Sie unter Abrufen und Optimieren der Ursachenanalyse.
Delta (�)	Eine Spalte, die den Metriktabellen hinzugefü gt wird, wenn die Ablaufverfolgungssätze A und B beide aktiv sind. In der Delta-Spalte wird der prozentuale Unterschied zwischen Ablaufver folgungssatz A und B berechnet

Reaktionszeitverteilung

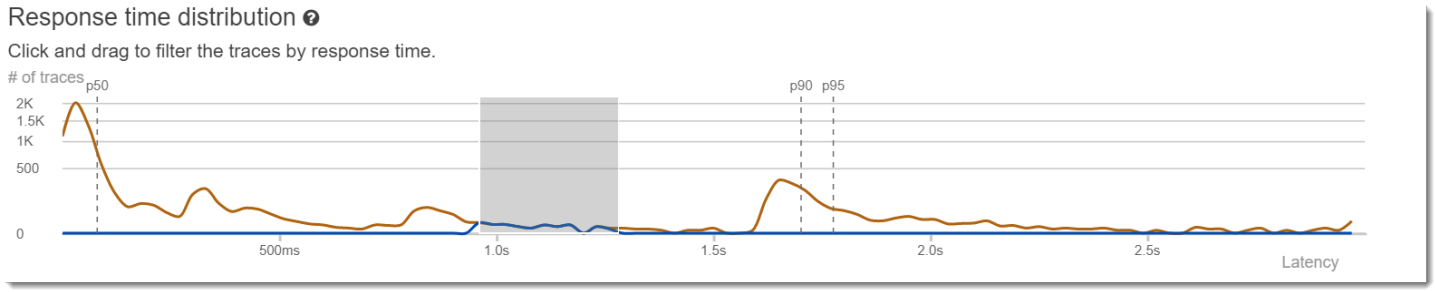
Die X-Ray Analytics-Konsole generiert zwei Hauptdiagramme, mit denen Sie Traces visualisieren können: Verteilung der Reaktionszeit und Zeitreihenaktivität. In diesem und den folgenden Abschnitten sehen Sie jeweils Beispiele dafür. Zudem werden die Grundlagen zum Lesen der Diagramme erläutert.

Im Folgenden sehen Sie die Farben, die dem Liniendiagramm „Reaktionszeit“ zugeordnet sind (das Zeitreihendiagramm verwendet dasselbe Farbschema):

- Alle Spuren in der Gruppe — grau
- Abgerufene Spuren — orange
- Gefilterter Trace-Satz A — grün
- Gefilterter Trace-Satz B — blau

Example— Verteilung der Antwortzeiten

Bei der Verteilung der Reaktionszeit handelt es sich um ein Diagramm, das die Anzahl der Ablaufverfolgungen innerhalb einer bestimmten Reaktionszeit zeigt. Klicken und ziehen Sie, um eine Auswahl innerhalb der Verteilung der Reaktionszeit zu treffen. Dadurch wird ein Filter im Arbeitssatz der Ablaufverfolgung mit der Bezeichnung `responseTime` für alle Ablaufverfolgungen innerhalb einer bestimmten Reaktionszeit ausgewählt und erstellt.

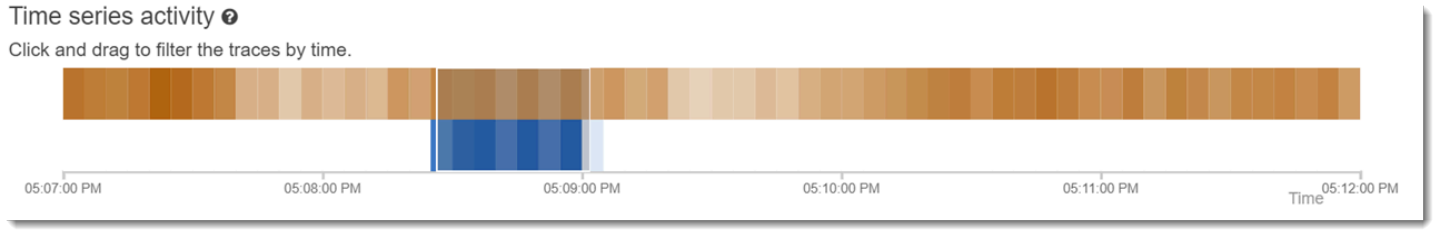


Zeitreihenaktivität

Das Diagramm „Zeitreihenaktivität“ zeigt die Anzahl der Ablaufverfolgungen in einem bestimmten Zeitraum. Die Farbindikatoren spiegeln die Farben im Liniendiagramm der Reaktionszeitverteilung wider. Je dunkler und voller der Farbblock innerhalb der Aktivitätsreihe, desto mehr Ablaufverfolgungen werden zum angegebenen Zeitpunkt dargestellt.

Example— Aktivität in Zeitreihen

Klicken und ziehen Sie, um im Diagramm „Zeitreihenaktivität“ eine Auswahl zu treffen. Dadurch wird ein Filter mit der Bezeichnung `timerange` im Arbeitssatz der Ablaufverfolgung für alle Ablaufverfolgungen innerhalb eines bestimmten Zeitraums ausgewählt und erstellt.



Workflow-Beispiele

Die folgenden Beispiele zeigen allgemeine Anwendungsfälle für die X-Ray Analytics-Konsole. Jedes Beispiel zeigt eine wichtige Funktion der Konsolenumgebung. Die Beispiele folgen als Gruppe einem grundlegenden Fehlerbehebungsworkflow. In den Schritten wird beschrieben, wie Sie zunächst nicht

betriebsbereite Knoten erkennen und dann mit der Analytics-Konsole interagieren, um vergleichende Abfragen automatisch zu generieren. Nachdem Sie den Bereich durch Abfragen eingegrenzt haben, sehen Sie sich abschließend die Details der für Sie relevanten Ablaufverfolgungen an, um zu ermitteln, was sich negativ auf die Integrität Ihres Services auswirkt.

Achten Sie auf Fehler auf im Servicediagramm.

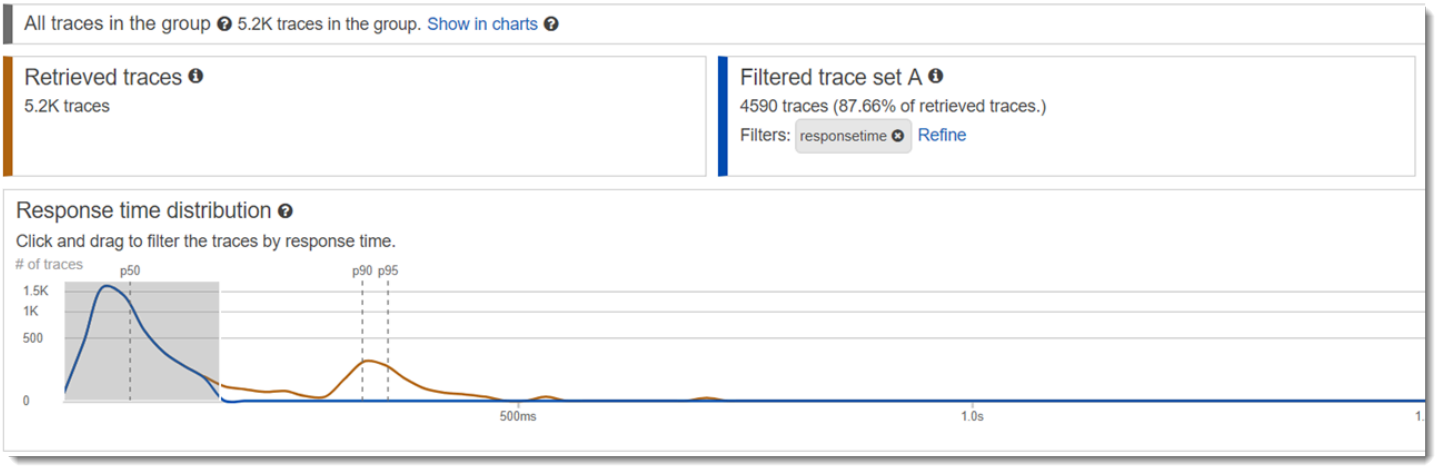
Die Trace-Map zeigt den Zustand der einzelnen Knoten an, indem sie sie anhand des Verhältnisses von erfolgreichen Aufrufen zu Fehlern und Störungen einfärbt. Ein roter Prozentsatz auf Ihrem Knoten signalisiert eine Störung. Verwenden Sie die X-Ray Analytics-Konsole, um das Problem zu untersuchen.

Weitere Informationen zum Lesen der Trace-Map finden Sie unter Trace-Map anzeigen.

Spitzen der Reaktionszeit erkennen

Mithilfe der Reaktionszeitverteilung können Sie Spitzen in der Reaktionszeit beobachten. Durch die Auswahl der Spitzen in der Reaktionszeit werden die Tabellen unterhalb der Diagramme aktualisiert und enthalten dann alle zugehörigen Metriken, z. B. die Statuscodes.

Wenn Sie klicken und ziehen, wählt X-Ray einen Filter aus und erstellt ihn. Er wird in einem grauen Schatten über den grau unterlegten Linien dargestellt. Sie können diesen Schatten jetzt entlang der Verteilung nach links und rechts ziehen, um die Auswahl und den Filter zu aktualisieren.



Alle Ablaufverfolgungen mit einem Statuscode anzeigen

Sie können Ablaufverfolgungen innerhalb der ausgewählten Spitze anhand der Metriktabellen unterhalb der Diagramme detailliert anzeigen. Durch Klicken auf eine Zeile in der Tabelle HTTP STATUS CODE (HTTP-STATUSCODE können Sie automatisch einen Filter im Arbeitsdatensatz erstellen. Beispiel: Sie können alle Ablaufverfolgungen für Statuscode 500 anzeigen. Dadurch wird ein Filter-Tag auf der Kachel „Ablaufverfolgungssatz“ mit dem Namen `http.status` erstellt.

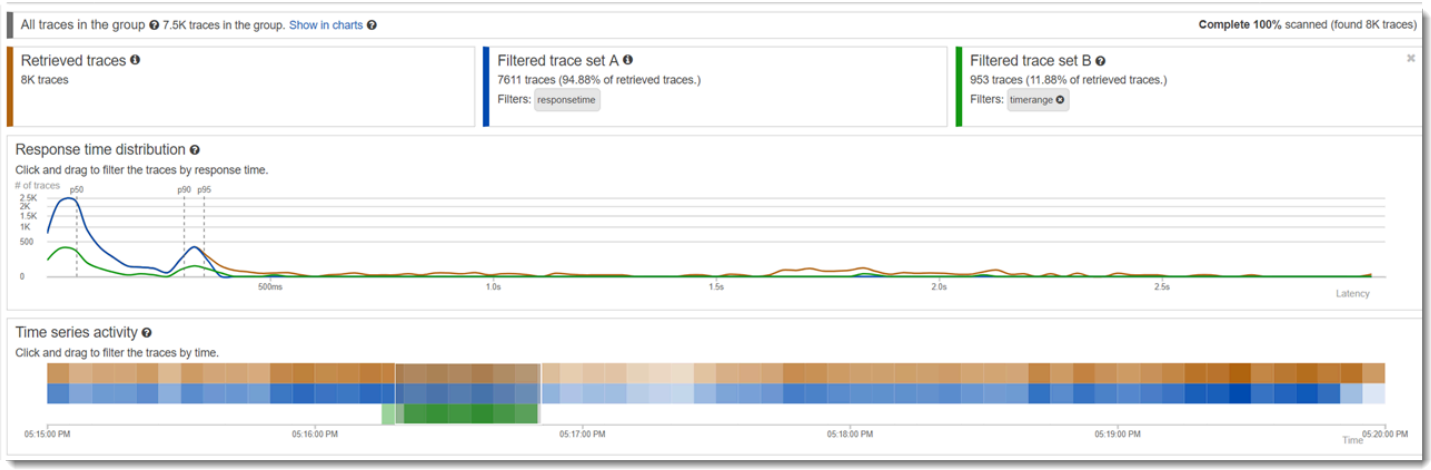
Alle Elemente in einer Untergruppe und einem Benutzer zugeordnete Elemente anzeigen

Führen Sie einen Drilldown in den Fehler auf Grundlage von Benutzer, URL, Ursache Reaktionszeit oder anderen vordefinierten Attributen durch. Wenn Sie beispielsweise den Satz der Ablaufverfolgungen mit Statuscode 500 filtern möchten, wählen Sie eine Zeile aus der Tabelle USERS (BENUTZER) aus. Dies führt zu zwei Filter-Tags auf der Kachel „Ablaufverfolgungssatz“: `http.status`, wie zuvor festgelegt, und `user`.

Vergleichen Sie zwei Ablaufverfolgungssätze mit unterschiedlichen Kriterien

Vergleichen Sie verschiedene Benutzer und ihre POST-Anfragen, um weitere Abweichungen und Korrelationen zu finden. Wenden Sie Ihren ersten Satz Filter an. Sie sind anhand einer blauen Linie in der Reaktionszeitverteilung definiert. Wählen Sie dann Compare (Vergleichen) aus. Zu Beginn wird dadurch eine Kopie der Filter im Ablaufverfolgungssatz A erstellt.

Um fortzufahren, definieren Sie einen neuen Satz Filter, die auf Ablaufverfolgungssatz B angewendet werden. Dieser zweite Satz wird durch eine grüne Linie dargestellt. Das folgende Beispiel zeigt die verschiedenen Linien entsprechend dem blauen und grünen Farbschema.



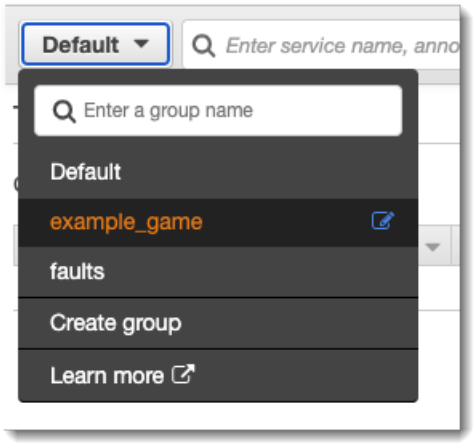
Zeigen Sie Details einer interessanten Ablaufverfolgung an

Wenn Sie mithilfe der Konsolenfilter den Umfang eingrenzen, erhält die Ablaufverfolgungsliste unterhalb der Metriktabellen mehr Bedeutung. Die Tabelle mit der Ablaufverfolgungsliste vereint Informationen über URL, USER (BENUTZER) und STATUS CODE (STATUSCODE) in einer Ansicht. Weitere Einblicke erhalten Sie, wenn Sie eine Zeile aus dieser Tabelle auswählen und so die Detailseite der Ablaufverfolgung öffnen und die Zeitleiste und Rohdaten anzeigen.

Gruppen konfigurieren

Gruppen sind eine Sammlung von Ablaufverfolgungen, die durch einen Filterausdruck definiert sind. Sie können Gruppen verwenden, um zusätzliche Servicegraphen zu erstellen und CloudWatch Amazon-Metriken bereitzustellen. Sie können die AWS X-Ray Konsole oder die X-Ray-API verwenden, um Gruppen für Ihre Dienste zu erstellen und zu verwalten. In diesem Thema wird beschrieben, wie Gruppen mithilfe der X-Ray-Konsole erstellt und verwaltet werden. Informationen zur Verwaltung von Gruppen mithilfe der X-Ray-API finden Sie unterGruppen.

Sie können Gruppen von Traces für Trace-Maps, Traces oder Analysen erstellen. Wenn Sie eine Gruppe erstellen, ist die Gruppe als Filter im Gruppen-Dropdownmenü auf allen drei Seiten verfügbar: Trace Map, Traces und Analytics.



Gruppen werden über ihren Namen oder einen Amazon-Ressourcennamen (ARN) identifiziert und enthalten einen Filterausdruck. Der Service vergleicht eingehende Ablaufverfolgungen mit dem Ausdruck und speichert sie entsprechend. Weitere Hinweise zum Erstellen eines Filterausdrucks finden Sie unterVerwenden von Filterausdrücken.

Durch das Aktualisieren des Filterausdrucks einer Gruppe werden die bereits aufgezeichneten Daten nicht geändert. Die Aktualisierung gilt nur für nachfolgende Ablaufverfolgungen. Dies kann dazu führen, dass im Diagramm neue und alte Ausdrücke zusammengeführt werden. Um dies zu vermeiden, löschen Sie eine aktuelle Gruppe und erstellen Sie eine neue.

> **Note**
>
> Gruppen werden anhand der Anzahl der abgerufenen Ablaufverfolgungen, die dem Filterausdruck entsprechen, in Rechnung gestellt. Weitere Informationen finden Sie unter AWS X-Ray Preise.

Themen

- Gruppenmetriken in Amazon anzeigen CloudWatch

Erstellen einer Gruppe

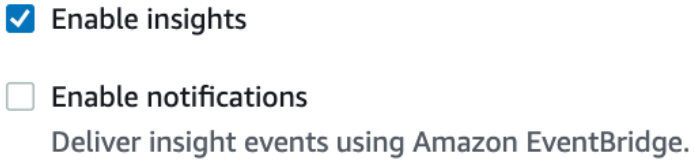
> **Note**
>
> Sie können jetzt X-Ray-Gruppen von der CloudWatch Amazon-Konsole aus konfigurieren. Sie können die X-Ray-Konsole auch weiterhin verwenden.

CloudWatch console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich Einstellungen aus.
3. Wählen Sie im Bereich X-Ray-Traces unter Gruppen die Option Einstellungen anzeigen.
4. Wählen Sie oberhalb der Gruppenliste die Option Gruppe erstellen aus.
5. Geben Sie auf der Seite Gruppe erstellen einen Namen für die Gruppe ein. Ein Gruppenname kann maximal 32 Zeichen lang sein und alphanumerische Zeichen und Bindestriche enthalten. Bei Gruppennamen wird zwischen Groß- und Kleinschreibung unterschieden.
6. Geben Sie einen Filterausdruck ein. Weitere Informationen zum Erstellen eines Filterausdrucks finden Sie unterVerwenden von Filterausdrücken. Im folgenden Beispiel filtert die Gruppe nach Fehlerspuren vom Dienst `api.example.com` und nach Anfragen an den Dienst, bei denen die Antwortzeit mindestens fünf Sekunden betrug.

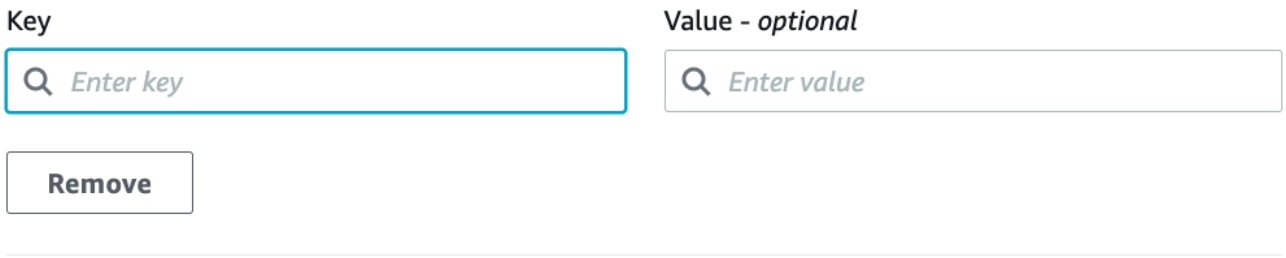
   ```
   fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
   ```

7. Aktivieren oder deaktivieren Sie in Insights den Insights-Zugriff für die Gruppe. Weitere Informationen über Funde sind unter Nutzung von X-Ray Insights verfügbar.

8. Wählen Sie unter Tags die Option Neues Tag hinzufügen aus, um einen Tag-Schlüssel und optional einen Tag-Wert einzugeben. Fügen Sie nach Bedarf weitere Tags hinzu. Tag-

Schlüssel müssen eindeutig sein. Um ein Tag zu löschen, wählen Sie unter jedem Tag die Option Entfernen aus. Weitere Informationen zu Tags erhalten Sie unter Kennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.



9. Wählen Sie Create group (Gruppe erstellen) aus.

X-Ray console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Öffnen Sie die Seite Gruppe erstellen über die Gruppenseite im linken Navigationsbereich oder über das Gruppenmenü auf einer der folgenden Seiten: Trace Map, Traces und Analytics.
3. Geben Sie auf der Seite Gruppe erstellen einen Namen für die Gruppe ein. Ein Gruppenname kann maximal 32 Zeichen lang sein und alphanumerische Zeichen und Bindestriche enthalten. Bei Gruppennamen wird zwischen Groß- und Kleinschreibung unterschieden.
4. Geben Sie einen Filterausdruck ein. Weitere Informationen zum Erstellen eines Filterausdrucks finden Sie unterVerwenden von Filterausdrücken. Im folgenden Beispiel filtert die Gruppe nach Fehlerspuren vom Dienst `api.example.com` und nach Anfragen an den Dienst, bei denen die Antwortzeit mindestens fünf Sekunden betrug.

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

5. Aktivieren oder deaktivieren Sie in Insights den Insights-Zugriff für die Gruppe. Weitere Informationen über Funde sind unter Nutzung von X-Ray Insights verfügbar.

6. Geben Sie im Feld Tags einen Tag-Schlüssel und optional einen Tag-Wert ein. Wenn Sie ein Tag hinzufügen, wird eine neue Zeile angezeigt, in der Sie ein anderes Tag eingeben können. Tag-Schlüssel müssen eindeutig sein. Um ein Tag zu löschen, wählen Sie X am Ende der

Tag-Zeile aus. Weitere Informationen zu Tags erhalten Sie unter Kennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

7. Wählen Sie Create group (Gruppe erstellen) aus.

Wenden Sie eine Gruppe an

CloudWatch console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Öffnen Sie im Navigationsbereich unter X-Ray Traces eine der folgenden Seiten:
   - Trace-Map
   - Ablaufverfolgungen
3. Geben Sie einen Gruppennamen in den Gruppenfilter Filter by X-Ray ein. Die auf der Seite angezeigten Daten ändern sich entsprechend dem in der Gruppe festgelegten Filterausdruck.

X-Ray console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Öffnen Sie im Navigationsbereich eine der folgenden Seiten:
   - Karte verfolgen
   - Ablaufverfolgungen
   - Analysen
3. Wählen Sie im Gruppenmenü die Gruppe aus, in der Sie erstellt habenthe section called “Erstellen einer Gruppe”. Die auf der Seite angezeigten Daten ändern sich entsprechend dem in der Gruppe festgelegten Filterausdruck.

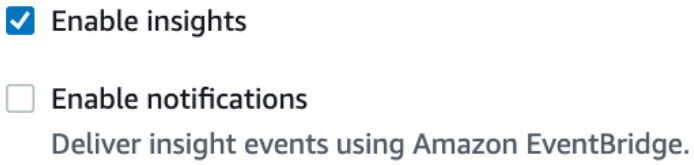
Bearbeiten Sie eine Gruppe

CloudWatch console

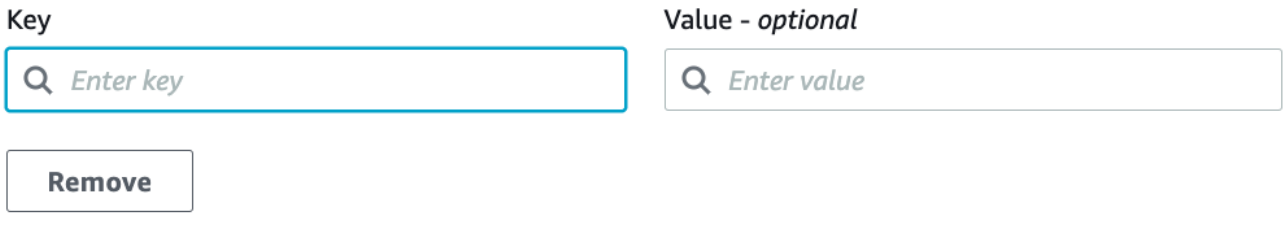
1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich Einstellungen aus.
3. Wählen Sie im Bereich X-Ray-Traces unter Gruppen die Option Einstellungen anzeigen.
4. Wählen Sie im Bereich Gruppen eine Gruppe aus und klicken Sie dann auf Bearbeiten.
5. Sie können eine Gruppe zwar nicht umbenennen, aber Sie können den Filterausdruck aktualisieren. Weitere Informationen zum Erstellen eines Filterausdrucks finden Sie unterVerwenden von Filterausdrücken. Im folgenden Beispiel filtert die Gruppe nach Fehlerablaufverfolgungen des Dienstes`api.example.com`, bei dem die URL-Adresse der Anfrage Folgendes enthält`example/game`: Die Antwortzeit für Anfragen betrug mindestens fünf Sekunden.

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

6. Aktivieren oder deaktivieren Sie in Insights den Insights-Zugriff für die Gruppe. Weitere Informationen über Funde sind unter Nutzung von X-Ray Insights verfügbar.

7. Wählen Sie unter Tags die Option Neues Tag hinzufügen aus, um einen Tag-Schlüssel und optional einen Tag-Wert einzugeben. Fügen Sie nach Bedarf weitere Tags hinzu. Tag-Schlüssel müssen eindeutig sein. Um ein Tag zu löschen, wählen Sie unter jedem Tag die Option Entfernen aus. Weitere Informationen zu Tags erhalten Sie unter Kennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

8. Wenn Sie mit dem Aktualisieren der Gruppe fertig sind, wählen Sie Gruppe aktualisieren.

X-Ray console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Gehen Sie wie folgt vor, um die Seite Gruppe bearbeiten zu öffnen.
   a. Wählen Sie auf der Seite Gruppen den Namen einer Gruppe aus, um sie zu bearbeiten.
   b. Zeigen Sie im Gruppenmenü auf einer der folgenden Seiten auf eine Gruppe, und wählen Sie dann Bearbeiten aus.
      - Karte verfolgen
      - Ablaufverfolgungen
      - Analysen
3. Sie können eine Gruppe zwar nicht umbenennen, aber Sie können den Filterausdruck aktualisieren. Weitere Informationen zum Erstellen eines Filterausdrucks finden Sie unterVerwenden von Filterausdrücken. Im folgenden Beispiel filtert die Gruppe nach Fehlerablaufverfolgungen des Dienstes`api.example.com`, bei dem die URL-Adresse der Anfrage Folgendes enthält`example/game`: Die Antwortzeit für Anfragen betrug mindestens fünf Sekunden.

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

4. Aktivieren oder deaktivieren Sie in Insights Insights und Insights-Benachrichtigungen für die Gruppe. Weitere Informationen über Funde sind unter Nutzung von X-Ray Insights verfügbar.

5. Bearbeiten Sie unter Tags die Tag-Schlüssel und -Werte. Tag-Schlüssel müssen eindeutig sein. Tag-Werte sind optional; Sie können Werte löschen, wenn Sie möchten. Um ein Tag zu löschen, wählen Sie X am Ende der Tagzeile aus. Weitere Informationen zu Tags erhalten Sie unter Kennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

6. Wenn Sie mit dem Aktualisieren der Gruppe fertig sind, wählen Sie Gruppe aktualisieren.

Eine Gruppe klonen

Durch das Klonen einer Gruppe wird eine neue Gruppe erstellt, die den Filterausdruck und die Tags einer vorhandenen Gruppe enthält. Wenn Sie eine Gruppe klonen, hat die neue Gruppe denselben Namen wie die Gruppe, aus der sie geklont wurde, wobei der Name an den Namen `-clone` angehängt wird.

CloudWatch console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter. https://console.aws.amazon.com/cloudwatch/
2. Wählen Sie im linken Navigationsbereich Einstellungen aus.
3. Wählen Sie im Bereich X-Ray-Traces unter Gruppen die Option Einstellungen anzeigen.
4. Wählen Sie im Bereich Gruppen eine Gruppe aus und klicken Sie dann auf Klonen.
5. Auf der Seite Gruppe erstellen lautet der Name der Gruppe *`group-name`*`-clone`. Geben Sie optional einen neuen Namen für die Gruppe ein. Ein Gruppenname kann maximal 32 Zeichen lang sein und alphanumerische Zeichen und Bindestriche enthalten. Bei Gruppennamen wird zwischen Groß- und Kleinschreibung unterschieden.
6. Sie können den Filterausdruck aus der vorhandenen Gruppe beibehalten oder optional einen neuen Filterausdruck eingeben. Weitere Informationen zum Erstellen eines Filterausdrucks finden Sie unterVerwenden von Filterausdrücken. Im folgenden Beispiel filtert die Gruppe nach Fehlerspuren vom Dienst `api.example.com` und nach Anfragen an den Dienst, bei denen die Antwortzeit mindestens fünf Sekunden betrug.

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

7. Bearbeiten Sie unter Tags die Tag-Schlüssel und -Werte, falls erforderlich. Tag-Schlüssel müssen eindeutig sein. Tag-Werte sind optional; Sie können Werte löschen, wenn Sie

möchten. Um ein Tag zu löschen, wählen Sie X am Ende der Tagzeile aus. Weitere Informationen zu Tags erhalten Sie unter Kennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

8. Wählen Sie Create group (Gruppe erstellen) aus.

X-Ray console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Öffnen Sie im linken Navigationsbereich die Gruppenseite und wählen Sie dann den Namen der Gruppe aus, die Sie klonen möchten.
3. Wählen Sie im Menü Aktionen die Option Gruppe klonen aus.
4. Auf der Seite Gruppe erstellen lautet der Name der Gruppe *`group-name`*`-clone`. Geben Sie optional einen neuen Namen für die Gruppe ein. Ein Gruppenname kann maximal 32 Zeichen lang sein und alphanumerische Zeichen und Bindestriche enthalten. Bei Gruppennamen wird zwischen Groß- und Kleinschreibung unterschieden.
5. Sie können den Filterausdruck aus der vorhandenen Gruppe beibehalten oder optional einen neuen Filterausdruck eingeben. Weitere Informationen zum Erstellen eines Filterausdrucks finden Sie unterVerwenden von Filterausdrücken. Im folgenden Beispiel filtert die Gruppe nach Fehlerspuren vom Dienst `api.example.com` und nach Anfragen an den Dienst, bei denen die Antwortzeit mindestens fünf Sekunden betrug.

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

6. Bearbeiten Sie unter Tags die Tag-Schlüssel und -Werte, falls erforderlich. Tag-Schlüssel müssen eindeutig sein. Tag-Werte sind optional; Sie können Werte löschen, wenn Sie möchten. Um ein Tag zu löschen, wählen Sie X am Ende der Tagzeile aus. Weitere Informationen zu Tags erhalten Sie unter Kennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.
7. Wählen Sie Create group (Gruppe erstellen) aus.

Löschen einer Gruppe

Folgen Sie den Schritten in diesem Abschnitt, um eine Gruppe zu löschen. Sie können die Standardgruppe nicht löschen.

CloudWatch console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich Einstellungen aus.
3. Wählen Sie im Bereich X-Ray-Traces unter Gruppen die Option Einstellungen anzeigen.
4. Wählen Sie im Bereich Gruppen eine Gruppe aus und klicken Sie dann auf Löschen.
5. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Löschen.

X-Ray console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Öffnen Sie im linken Navigationsbereich die Gruppenseite und wählen Sie dann den Namen der Gruppe aus, die Sie löschen möchten.
3. Wählen Sie im Menü Aktionen die Option Gruppe löschen aus.
4. Wenn Sie zur Bestätigung aufgefordert werden, wählen Sie Löschen.

Gruppenmetriken in Amazon anzeigen CloudWatch

Nachdem eine Gruppe erstellt wurde, werden eingehende Traces mit dem Filterausdruck der Gruppe verglichen, während sie im X-Ray-Dienst gespeichert werden. Metriken für die Anzahl der Traces, die den einzelnen Kriterien entsprechen, werden CloudWatch jede Minute auf Amazon veröffentlicht. Wenn Sie auf der Seite Gruppe bearbeiten die Option Metrik anzeigen wählen, wird in der CloudWatch Konsole die Seite Metrik geöffnet. Weitere Informationen zur Verwendung von CloudWatch Metriken finden Sie unter Using Amazon CloudWatch Metrics im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich Einstellungen aus.
3. Wählen Sie im Bereich X-Ray-Traces unter Gruppen die Option Einstellungen anzeigen.
4. Wählen Sie im Bereich Gruppen eine Gruppe aus und klicken Sie dann auf Bearbeiten.

5. Wählen Sie auf der Seite „Gruppe bearbeiten“ die Option Metrik anzeigen aus.

   Die Seite Metriken der CloudWatch Konsole wird auf einer neuen Registerkarte geöffnet.

X-Ray console

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Öffnen Sie im linken Navigationsbereich die Gruppenseite und wählen Sie dann den Namen einer Gruppe aus, für die Sie Messwerte anzeigen möchten.
3. Wählen Sie auf der Seite „Gruppe bearbeiten“ die Option Metrik anzeigen aus.

   Die Seite Metriken der CloudWatch Konsole wird auf einer neuen Registerkarte geöffnet.

Konfigurieren von -Samplingregeln

Sie können die AWS X-Ray Konsole verwenden, um Sampling-Regeln für Ihre Dienste zu konfigurieren. Das AWS Distro for OpenTelemetry, das X-Ray SDK und AWS-Services das Active Tracing mit Sampling-Konfigurationen unterstützen, verwenden Sampling-Regeln, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen.

Topics

Konfigurieren von -Samplingregeln

Sie können Sampling für die folgenden Anwendungsfälle konfigurieren:

- API Gateway Entrypoint — API Gateway unterstützt Sampling und Active Tracing. Informationen zum Aktivieren der aktiven Ablaufverfolgung für eine API-Stufe finden Sie unter Unterstützung für aktives Tracing von Amazon API Gateway für AWS X-Ray.
- AWS AppSync— AWS AppSync unterstützt Sampling und aktives Tracing. Informationen zum Aktivieren der aktiven Ablaufverfolgung für AWS AppSync Anfragen finden Sie unter Tracing with AWS X-Ray.
- AWS Step Functions— AWS Step Functions unterstützt Sampling und aktives Tracing. Informationen zur Aktivierung der aktiven Ablaufverfolgung auf AWS Step Functions Zustandsmaschinen finden Sie unter X-Ray Tracing in Step Functions.
- Instrument AWS Distro for OpenTelemetry on Compute-Plattformen — Bei der Verwendung von Rechenplattformen wie Amazon EC2, Amazon ECS oder wird Sampling unterstützt AWS Elastic Beanstalk, wenn die Anwendung mit dem neuesten AWS Distro for OpenTelemetry - oder X-Ray-SDK instrumentiert wurde.

Anpassen von Samplingregeln

Durch die Anpassung der Stichprobenregeln können Sie die Menge der aufgenommenen Daten kontrollieren. Sie können auch das Sampling-Verhalten ändern, ohne Ihren Code zu ändern oder erneut bereitzustellen. Sampling-Regeln teilen dem AWS Distro for OpenTelemetry (ADOT) oder X-Ray SDK mit, wie viele Anfragen für eine Reihe von Kriterien aufgezeichnet werden müssen. Standardmäßig zeichnet das SDK jede Sekunde die erste Anfrage und fünf Prozent aller weiteren Anfragen auf. Eine Anfrage pro Sekunde ist das Reservoir. Dadurch wird sichergestellt, dass jede Sekunde mindestens eine Ablaufverfolgung aufgezeichnet wird, solange der Dienst Anfragen verarbeitet. Fünf Prozent ist die Rate, mit der die über die Reservoirgröße hinausgehenden Anforderungen geprüft werden.

Sie können das X-Ray SDK so konfigurieren, dass es Sampling-Regeln aus einem JSON-Dokument liest, das Sie in Ihren Code einbinden. Wenn Sie jedoch mehrere Instances Ihres Services ausführen, führt jede Instance das Sampling unabhängig aus. Dies bewirkt, dass sich der Gesamtprozentsatz der geprüften Anforderungen erhöht, da die Reservoirs aller Instances effektiv zusammengezählt werden. Um lokale Sampling-Regeln zu aktualisieren, müssen Sie Ihren Code außerdem erneut bereitstellen.

Indem Sie Sampling-Regeln in der X-Ray-Konsole definieren und das SDK so konfigurieren, dass es Regeln aus dem X-Ray-Dienst liest, können Sie beide Probleme vermeiden. Der Service verwaltet die Reservoirs für jede Regel und weist jeder Instance Ihres Services Kontingente zu, um das Reservoir gleichmäßig zu verteilen, basierend auf der Anzahl der Instances, die ausgeführt werden. Das

Reservoir-Limit wird gemäß den von Ihnen festgelegten Regeln berechnet. Da die Regeln im Dienst konfiguriert sind, können Sie Regeln verwalten, ohne zusätzliche Implementierungen vornehmen zu müssen.

> **Note**
>
> Bei der Konfiguration von Probenahmeregeln ist es wichtig zu wissen, dass die Röntgenprobenentnahme „übergeordnetes System“ ist. Das bedeutet, dass die Stichprobenentscheidung nur einmal getroffen wird, in der Regel vom ersten X-Ray-enabled Dienst, der die Anfrage bearbeitet (der „Root“ -Dienst).
> Wenn ein nachgelagerter Dienst eine Anfrage erhält, für die bereits eine Stichprobenentscheidung von einem übergeordneten Dienst getroffen wurde, wird er diese Entscheidung unabhängig von seinen eigenen Stichprobenregeln berücksichtigen.
>
> - Wenn Regeln gelten: Ihre benutzerdefinierten Stichprobenregeln gelten nur für Dienste, für die noch keine Stichprobenentscheidung getroffen wurde. Dies gilt normalerweise für:
>   - Der Einstiegspunkt Ihrer Anwendung (z. B. ein API Gateway, ein Load Balancer oder der erste instrumentierte Microservice).
>   - Asynchrone Prozesse oder Worker, die einen brandneuen Trace starten.
> - Häufiger Fallstrick: Wenn Sie eine strenge Stichprobenregel für „Service B“ erstellen, „Service B“ aber immer von „Service A“ aufgerufen wird, wird Ihre Regel für Service B wahrscheinlich nie angewendet, da sie einfach der Entscheidung von Service A folgt.
> Um die Stichprobenziehung von Traces für diesen Workflow zu ändern, müssen Sie die Stichprobenregel für den Root-Dienst (Service A) konfigurieren.

> **Note**
>
> X-Ray verwendet bei der Anwendung von Stichprobenregeln einen Best-Effort-Ansatz, und in einigen Fällen entspricht die effektive Abtastrate möglicherweise nicht genau den konfigurierten Abtastregeln. Im Laufe der Zeit sollte die Anzahl der gesampelten Anfragen jedoch in der Nähe des konfigurierten Prozentsatzes liegen.

Sie können jetzt X-Ray-Sampling-Regeln von der CloudWatch Amazon-Konsole aus konfigurieren.

Um Sampling-Regeln in der CloudWatch Konsole zu konfigurieren

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich unter Setup die Option Einstellungen aus.
3. Wählen Sie im Bereich X-Ray-Traces unter Sampling-Regeln die Option Einstellungen anzeigen.
4. Um eine Regel zu erstellen, wählen Sie Create sampling rule (Samplingregel erstellen) aus.

   Um eine Regel zu bearbeiten, wählen Sie eine Regel aus und klicken Sie auf Bearbeiten, um sie zu bearbeiten.

   Um eine Regel zu löschen, wählen Sie eine Regel aus und wählen Sie Löschen, um sie zu löschen.

Optionen für Samplingregeln

Für jede Regel stehen folgende Optionen zur Verfügung. Für Zeichenfolgenwerte können Platzhalter verwendet werden, um einem einzelnen Zeichen (?) oder null oder mehreren Zeichen (*) zu entsprechen.

Optionen für Samplingregeln

- Regelname (Zeichenfolge) — Ein eindeutiger Name für die Regel.
- Priorität (Ganzzahl zwischen 1 und 9999) — Die Priorität der Stichprobenregel. Services werten Regeln in aufsteigender Reihenfolge der Priorität aus und treffen eine Sampleentscheidung mit der ersten übereinstimmenden Regel.
- Reservoir (nicht negative Ganzzahl) — Eine feste Anzahl von Abgleichsanfragen an das Instrument pro Sekunde, bevor die feste Rate angewendet wird. Das Reservoir wird nicht direkt von Services verwendet, sondern gilt für alle Services, die die Regel gemeinsam verwenden.
- Rate (Ganzzahl zwischen 0 und 100) — Der Prozentsatz der übereinstimmenden Anfragen an das Instrument, nachdem das Reservoir aufgebraucht ist. Wählen Sie bei der Konfiguration einer Stichprobenregel in der Konsole einen Prozentsatz zwischen 0 und 100. Geben Sie bei der Konfiguration einer Stichprobenregel in einem Client-SDK mithilfe eines JSON-Dokuments einen Prozentwert zwischen 0 und 1 an.
- Dienstname (Zeichenfolge) — Der Name des instrumentierten Dienstes, wie er in der Trace-Map angezeigt wird.

  - X-Ray SDK — Der Dienstname, den Sie auf dem Rekorder konfigurieren.
  - Amazon API Gateway —*api-name*/*stage*.
- Servicetyp (Zeichenfolge) — Der Servicetyp, wie er in der Trace-Map angezeigt wird. Legen Sie für das X-Ray SDK den Diensttyp fest, indem Sie das entsprechende Plugin anwenden:
  - `AWS::ElasticBeanstalk::Environment`— Eine AWS Elastic Beanstalk Umgebung (Plugin).
  - `AWS::EC2::Instance`— Eine EC2 Amazon-Instanz (Plugin).
  - `AWS::ECS::Container`— Ein Amazon ECS-Container (Plugin).
  - `AWS::EKS::Container`— Ein Amazon EKS-Container (Plugin).
  - `AWS::APIGateway::Stage`— Eine Amazon API Gateway Gateway-Phase.
  - `AWS::AppSync::GraphQLAPI` — Eine AWS AppSync API-Anfrage.
  - `AWS::StepFunctions::StateMachine`— Eine AWS Step Functions Zustandsmaschine.
- Host (string) — Der Hostname aus dem HTTP-Host-Header.
- HTTP-Methode (Zeichenfolge) — Die Methode der HTTP-Anfrage.
- URL-Pfad (Zeichenfolge) — Der URL-Pfad der Anfrage.
  - X-Ray SDK — Der Pfadteil der HTTP-Anforderungs-URL.
- Ressourcen-ARN (Zeichenfolge) — Der ARN der AWS Ressource, auf der der Dienst ausgeführt wird.
  - X-Ray SDK — Nicht unterstützt. Das SDK kann nur Regeln verwenden, bei denen der Ressourcen-ARN auf * festgelegt ist.
  - Amazon API Gateway — Die Stufe ARN.
- (Optional) Attribute (Schlüssel und Wert) — Segmentattribute, die bekannt sind, als die Stichprobenentscheidung getroffen wurde.
  - X-Ray SDK — Nicht unterstützt. Das SDK ignoriert Regeln, die Attribute angeben.
  - Amazon API Gateway — Header aus der ursprünglichen HTTP-Anfrage.
- (Optional) SamplingRateBoost(Objekt) — Steuert das anomaliebedingte Boost-Verhalten bei der Probennahme.
  - MaxRate (Ganzzahl zwischen 0 und 100) — Die maximale Probenahmerate (0,0—1,0) X-Ray bei Anomalien bis zu einem Grad ansteigen
  - CooldownWindowMinutes (Ganzzahl größer als 0) — Zeitfenster (Minuten), in dem nur ein Boost ausgelöst werden kann, wodurch kontinuierliche Boosts verhindert werden

Beispiele für Samplingregeln

Example— Standardregel ohne Reservoir und niedriger Rate

Sie können das Reservoir und die Rate der Standardregel ändern. Die Standardregel gilt für alle Anforderungen, die nicht mit einer anderen Regel übereinstimmen.

- Reservoir: **`0`**
- Rate: **`5`** (**`0.05`**falls mit einem JSON-Dokument konfiguriert)

Example— Debugging-Regel zur Verfolgung aller Anfragen für eine problematische Route

Eine Regel mit hoher Priorität, die vorübergehend für das Debuggen angewendet wird.

- Name der Regel: **`DEBUG - history updates`**
- Priorität: **`1`**
- Reservoir: **`1`**
- Rate: **`100`** (**1**falls mit einem JSON-Dokument konfiguriert)
- Name des Dienstes: **`Scorekeep`**
- Service type (Servicetyp): *****
- Gastgeber: *****
- HTTP-Methode: **`PUT`**
- URL-Pfad: **`/history/*`**
- Ressourcen-ARN: *****

Example— Höherer Mindestsatz für POSTs

- Name der Regel: **`POST minimum`**
- Priorität: **`100`**
- Reservoir: **`10`**
- Rate: **`10`** (**`.1`**falls mit einem JSON-Dokument konfiguriert)
- Name des Dienstes: *****
- Service type (Servicetyp): *****
- Gastgeber: *****

- HTTP-Methode: **POST**
- URL-Pfad: ***
- Ressourcen-ARN: ***

Example Aktivieren Sie den durch Anomalien verursachten Boost

Konfiguriere eine Regel, die bei Anomalien einen Sampling-Boost von bis zu 50% mit einer Abklingzeit von 10 Minuten auslöst.

- Name der Regel: **MyAdaptiveRule**
- Priorität: **100**
- Reservoir: **1**
- FixedRate: **0.0510**
- Name des Dienstes: ***
- Service type (Servicetyp): ***
- Gastgeber: ***
- HTTP-Methode: **POST**
- URL-Pfad: ***
- maximale Rate: **0.5**
- cooldownWindowMinutes: **10**

Konfigurieren Ihres Service für die Verwendung von Samplingregeln

Das SDK für AWS Distro for OpenTelemetry (ADOT) und X-Ray erfordert eine zusätzliche Konfiguration, um Sampling-Regeln verwenden zu können, die Sie in der Konsole konfigurieren. Im Konfigurationsthema finden Sie in Ihrer Sprache Einzelheiten zur Konfiguration einer Samplingstrategie:

- Java: Verwendung von X-Ray Remote Sampling mit ADOT Java
- Go: Sampling mit ADOT Go konfigurieren
- Node.js: Verwendung von X-Ray Remote Sampling mit ADOT JavaScript
- Python: Verwendung von X-Ray Remote Sampling mit ADOT Python
- Rubin: Regeln für die Probenahme

- .NET: Verwendung von X-Ray Remote Sampling mit ADOT .NET

Informationen zu API Gateway finden Sie unterUnterstützung für aktives Tracing von Amazon API Gateway für AWS X-Ray.

Anzeigen von Samplingergebnissen

Auf der Sampling-Seite der X-Ray-Konsole finden Sie detaillierte Informationen darüber, wie Ihre Dienste die einzelnen Sampling-Regeln verwenden.

Die Spalte Trend zeigt, wie die Regel in den letzten paar Minuten verwendet wurde. Jede Spalte zeigt Statistiken für ein 10-Sekunden-Fenster an.

Samplingstatistiken

- Gesamtzahl der übereinstimmenden Regeln: Die Anzahl der Anfragen, die dieser Regel entsprachen. Diese Zahl umfasst keine Anforderungen, die mit dieser Regel übereingestimmt hätten, aber zuvor mit einer Regel mit höherer Priorität übereingestimmt haben.
- Gesamtzahl der gesammelten Anfragen: Die Anzahl der aufgezeichneten Anfragen.
- Stichprobe mit fester Rate: Die Anzahl der Anfragen, bei denen die feste Rate der Regel angewendet wurde.
- Mit Reservoirlimit abgetastet: Die Anzahl der Anfragen, die unter Verwendung einer von X-Ray zugewiesenen Quote abgetastet wurden.
- Aus dem Reservoir ausgeliehen: Die Anzahl der Anfragen, die durch Entleihen aus dem Reservoir entnommen wurden. Wenn ein Service eine Anfrage zum ersten Mal einer Regel zuordnet, wurde ihm von X-Ray noch kein Kontingent zugewiesen. Wenn das Reservoir jedoch mindestens 1 ist, leiht sich der Dienst eine Spur pro Sekunde aus, bis X-Ray eine Quote zuweist.

Weitere Informationen zu Samplingstatistiken und dazu, wie Services Samplingregeln verwenden, finden Sie unter Verwenden von Sampling-Regeln mit der X-Ray-API.

Nächste Schritte

Sie können die X-Ray-API verwenden, um Sampling-Regeln zu verwalten. Mit der API können Sie Regeln erstellen und aktualisieren, und zwar programmgesteuert nach einem Zeitplan oder als Reaktion auf Alarme oder Benachrichtigungen. Anleitungen und weitere Regelbeispiele finden Sie unter Konfigurieren von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray -API.

Das AWS Distro for OpenTelemetry, das X-Ray SDK und AWS-Services die X-Ray-API verwenden, um Probenahmeregeln zu lesen, Probenahmeergebnisse zu melden und Probenahmeziele abzurufen. Services müssen verfolgen, wie oft sie die einzelnen Regeln anwenden, Regeln anhand ihrer Priorität auswerten und Daten aus dem Reservoir abrufen, wenn eine Anfrage mit einer Regel übereinstimmt, für die X-Ray dem Service noch kein Kontingent zugewiesen hat. Weitere Informationen dazu, wie ein Service die API für das Sampling verwendet, finden Sie unter Verwenden von Sampling-Regeln mit der X-Ray-API.

Wenn die AWS Distribution for OpenTelemetry oder das X-Ray SDK Sampling aufrufen APIs, verwenden sie den CloudWatch Agenten als Proxy. Wenn Sie bereits den TCP-Port 2000 verwenden, können Sie den Agenten so konfigurieren, dass er den Proxy auf einem anderen Port ausführt. Einzelheiten finden Sie in der Installationsanleitung für den CloudWatch Agenten.

Konfiguration der adaptiven Probenahme

Das Fehlen kritischer Spuren bei Anomaliespitzen kann die Ursachenanalyse erschweren. Die Aufrechterhaltung hoher Probenraten ist jedoch teuer. Die adaptive Röntgenprobenentnahme bietet einen vollständigen Überblick über Anomalien und kontrolliert die Kosten während des normalen Betriebs. Bei der adaptiven Abtastung legen Sie eine maximale Abtastrate fest, und X-Ray passt sich automatisch innerhalb dieser Grenze an. X-Ray berechnet den minimalen Boost, der zur Erfassung von Fehlerspuren erforderlich ist. Wenn Ihre Basisrate genügend Daten erfasst, erfolgt kein Boost. Sie zahlen nur für zusätzliche Probenahmen, wenn sie benötigt werden.

Vorteile der adaptiven Probenahme:

- Vollständige Transparenz bei Vorfällen — Verschaffen Sie sich vollständige Nachverfolgbarkeit von Vorfällen ohne manuelles Eingreifen. X-Ray passt die Abtastraten automatisch an, um Fehlerspuren zu erfassen, und kehrt dann zu normalen Raten zurück.
- Sichtbarkeit der Ursache — Erkennen Sie stets die Ursache von Problemen. X-Ray erfasst kritische Fehlerdaten, auch wenn keine vollständige Spurenprobenahme ausgelöst wird.
- Kosten optimieren — Kurzfristige Probenahmen (bis zu 1 Minute) und automatische Abklingzeiten verhindern eine Überprobennahme. Sie zahlen nur für die Daten, die Sie zur Problemdiagnose benötigen.

Themen

- Lokale SDK-Konfiguration

Unterstützte Plattformen SDKs und Plattformen

Unterstütztes SDK — Adaptive Sampling erfordert die neueste Version des ADOT SDK.

Unterstützte Sprache — Java (Version v2.11.5 oder höher)

Ihre Anwendung muss mit dem unterstützten ADOT SDK instrumentiert und entweder zusammen mit dem Amazon CloudWatch Agent oder dem OpenTelemetry Collector ausgeführt werden.

Amazon EC2, Amazon ECS und Amazon EKS sind beispielsweise gängige Plattformen, auf denen AWS Application Signals Anleitungen zur Aktivierung des ADOT SDK und Amazon CloudWatch Agent bereitstellt.

Wählen Sie Ihren adaptiven Sampling-Ansatz

Adaptives Sampling unterstützt zwei Ansätze: Sampling Boost und Anomaly Span Capture. Diese können unabhängig voneinander angewendet oder miteinander kombiniert werden.

Steigerung der Probenahme

Adaptive Sampling Boost basiert auf Sampling-Regeln und funktioniert mit dem bestehenden Kopf-basierten Röntgen-Sampling-Modell. Bei der Stichprobenerhebung werden Entscheidungen über die Stichprobenerhebung beim Stammdienst getroffen und das Sampling-Flag anschließend an alle Dienste in der Aufrufkette weitergegeben.

- Regelbasiertes Boosten — Boosting ist immer an eine bestimmte Röntgen-Sampling-Regel gebunden. Jede Regel kann ihre eigene maximale Boost-Rate und ihr eigenes Abkühlverhalten definieren.
- Kopf-basiertes Sampling — Entscheidungen zur Probenahme werden beim Root-Service getroffen, und das Sampling-Flag wird nachgelagert an alle Services in der Aufrufkette weitergegeben.
- Anomaliegesteuert — X-Ray verwendet das SDK, um Anomaliestatistiken zu melden. Wenn X-Ray Anomalien wie Fehler oder hohe Latenz erkennt, verwendet es diese Statistiken, um eine angemessene Boost-Rate (bis zum konfigurierten Maximum) zu berechnen.

Meldung von Anomalien

Jeder Anwendungsdienst in der Aufrufkette kann über das erforderliche SDK Anomaliestatistiken ausgeben:

- Root-Dienst — Muss auf einem unterstützten SDK und einer unterstützten Plattform ausgeführt werden, um den Sampling-Boost zu aktivieren. Wenn der Root-Dienst nicht unterstützt wird, erfolgt kein Boost.
- Downstream-Dienste — Downstream-Dienste melden nur Anomalien; sie können keine Stichprobenentscheidungen treffen. Wenn auf einem nachgelagerten Dienst ein unterstütztes SDK ausgeführt wird, können festgestellte Anomalien einen Anstieg der Probenahme auslösen. Wenn ein nachgelagerter Dienst nicht unterstützt wird (z. B. wenn ein älteres SDK ausgeführt wird), lösen Anomalien bei diesem Dienst keinen Boost aus. Diese Dienste können den Kontext immer noch flussabwärts weitergeben, wenn sie der Standardweiterleitung des Kontextes folgen (z. B. W3C-Trace-Kontext und Baggage). Dadurch wird sichergestellt, dass die SDKs in weiteren nachgelagerten Diensten unterstützten Dienste Anomalien melden können, die einen Boost auslösen.

Erhöhen Sie den Zeitplan und den Umfang

- Triggerverzögerung — Sie können davon ausgehen, dass ein Sampling-Boost bereits 10 Sekunden nach der Erkennung einer Anomalie durch X-Ray einsetzt.
- Boost-Periode — Nachdem X-Ray einen Boost ausgelöst hat, hält er bis zu 1 Minute an, bevor er wieder zur Basis-Samplingrate zurückkehrt.
- Boost-Abkühlung — Nach einem Boost löst X-Ray keinen weiteren Boost für dieselbe Regel aus, bis das Abkühlfenster abgelaufen ist.

  Wenn du zum Beispiel `cooldown` auf 10 Minuten festlegst und ein Boost endet, kann bis zum nächsten 10-Minuten-Fenster kein neuer Boost ausgelöst werden.

  Sonderfall: Wenn du den Wert `cooldown` auf 1 Minute festlegst und ein Boost selbst bis zu 1 Minute andauern kann, können Boosts effektiv kontinuierlich ausgelöst werden, wenn die Anomalie weiterhin besteht.

> **Note**
>
> Verwenden Sie unterstützte Plattformen SDKs und Plattformen für Ihren Root-Service. Sampling Boost funktioniert nur mit unterstützten SDKs Plattformen. Sampling Boost hat zwar eine hohe Wahrscheinlichkeit, Spuren von Anomalien zu erfassen, es kann jedoch sein, dass nicht alle Spuren von Anomalien erfasst werden.

Erhöhen Sie die Sichtbarkeit

Wenn eine Sampling-Regel mit adaptivem Sampling-Boost konfiguriert ist, gibt X-Ray automatisch ausgelieferte Metriken aus, mit denen Sie die Boost-Aktivität überwachen können.

- Name der Metrik — `SamplingRate`
- Dimension — `RuleName` (auf den tatsächlichen Regelnamen gesetzt)

Jede Regel, für die diese `SamplingRateBoost` Option aktiviert ist, veröffentlicht ihre effektive Samplingrate, einschließlich der Basisrate und aller temporären Boosts. Das ermöglicht Ihnen Folgendes:

- Verfolge, wann Boosts ausgelöst werden
- Überwachen Sie die effektive Stichprobenrate für jede Regel
- Korrelieren Sie Boosts mit Anwendungsanomalien (wie Fehlerspitzen oder Latenzereignissen)

Sie können diese Metriken in Amazon CloudWatch Metrics unter dem Namespace AWS/X-Ray einsehen. Der Metrikwert ist eine Gleitkommazahl zwischen 0 und 1, die die effektive Stichprobenrate darstellt.

Konfigurieren Sie den Sampling-Boost mithilfe der Regeln für die X-Ray

Sie können die adaptive Probenahme direkt in Ihren bestehenden Regeln für die Röntgenabtastung aktivieren, indem Sie ein neues `SamplingRateBoost` Feld hinzufügen. Weitere Informationen finden Sie unter Sampling-Regeln anpassen. Dies bietet eine zentrale Möglichkeit, adaptives Sampling zu ermöglichen, ohne den Anwendungscode zu ändern oder die Anwendungsbereitstellung anzuwenden. Wenn Sie adaptives Sampling aktivieren, erhöht X-Ray automatisch die Abtastung bei Anomalien wie Fehlerspitzen oder Latenzausreißern und hält gleichzeitig die Abtastraten innerhalb des konfigurierten Maximums. `SamplingRateBoost`kann auf jede benutzerdefinierte Stichprobenregel mit Ausnahme der Stichprobenregel angewendet werden. `Default`

Das `SamplingRateBoost` Feld definiert die Obergrenze und das Verhalten bei anomaliebedingter Probenahme.

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

```
}
```

Der `MaxRate` definiert die maximale Abtastrate, mit der X-Ray Anomalien erkennt. Der Wertebereich ist `0.0` bis. `1.0` `"MaxRate": 0.25`Ermöglicht es beispielsweise, während eines Zeitfensters mit Anomalien die Anzahl der Anfragen anhand von Stichproben um bis zu 25% zu erhöhen. X-Ray bestimmt je nach Anomalieaktivität die angemessene Rate zwischen Ihrem Ausgangswert und dem Maximum.

Das `CooldownWindowMinutes` definiert ein Zeitfenster (in Minuten), in dem nur eine Erhöhung der Abtastrate ausgelöst werden kann. Nachdem ein Boost erfolgt ist, sind bis zum nächsten Fenster keine weiteren Boosts mehr zulässig. Der Wertetyp ist eine Ganzzahl (Minuten).

Beispielregel mit adaptiver Stichprobe

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

In diesem Beispiel beträgt die Ausgangsstichprobe 5% (`FixedRate: 0.05`). Bei Anomalien kann die Probenahme durch X-Ray um bis zu 25% erhöht werden ()`MaxRate: 0.25`. Boost nur einmal alle 10 Minuten.

Konfiguration des Zustands von Anomalien

Wenn keine Konfiguration für Anomaliebedingungen angegeben ist, verwendet das ADOT SDK HTTP 5xx-Fehlercodes als Standardbedingung für Anomalien, um den Sampling-Boost auszulösen.

Sie können Anomaliebedingungen auch lokal im unterstützten ADOT SDK mithilfe von Umgebungsvariablen fein abstimmen. Weitere Informationen finden Sie unter Lokale SDK-Konfiguration.

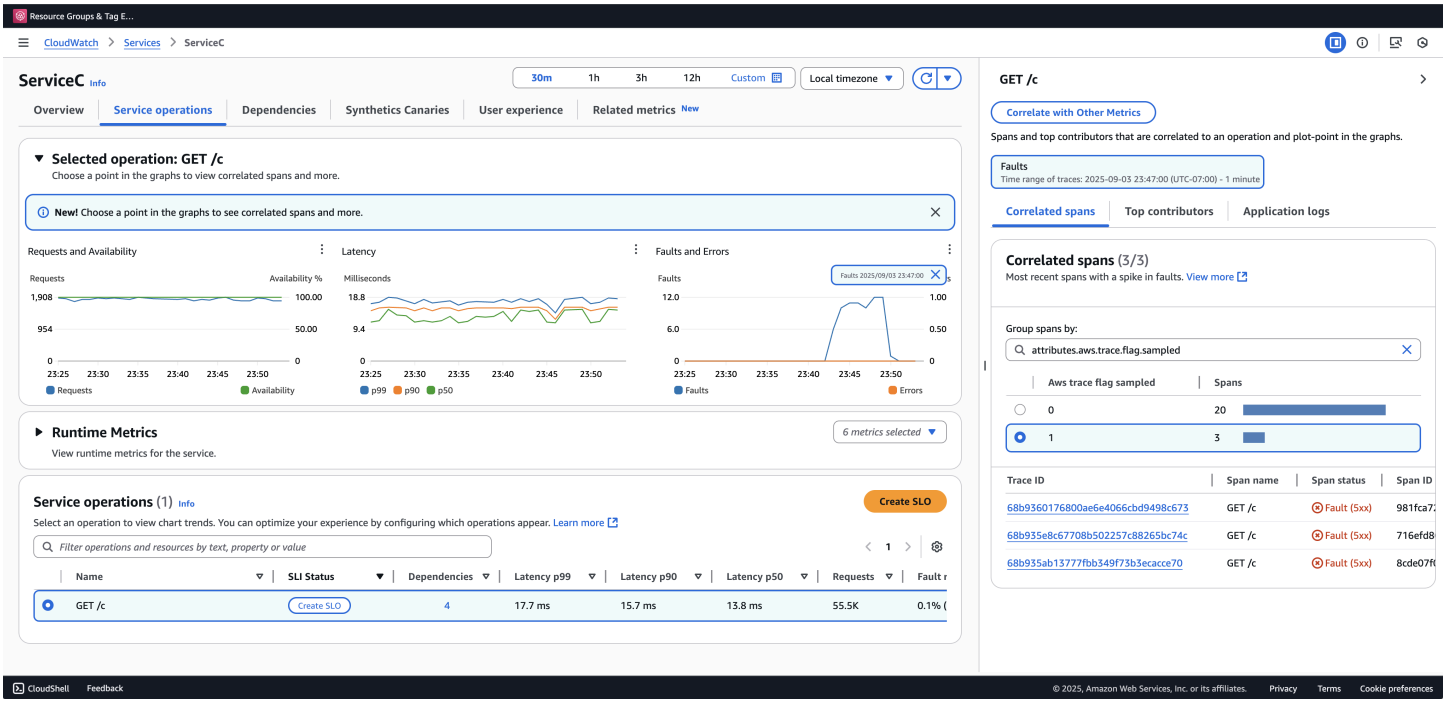
Anomalie erstreckt sich über die Erfassung

Durch die Erfassung von Anomaliebereichen wird sichergestellt, dass kritische Bereiche, die Anomalien darstellen, immer aufgezeichnet werden, auch wenn nicht die gesamte Spur erfasst wird. Diese Funktion ergänzt die Steigerung der Probenahme, indem sie sich auf die Erfassung der Anomalie selbst konzentriert, anstatt die Probenahme für future Spuren zu erhöhen.

Wenn das ADOT SDK eine Anomalie erkennt, gibt es diese Spanne sofort aus, unabhängig von der Entscheidung für die Probenahme. Da das SDK nur Spans ausgibt, die sich auf die Anomalie beziehen, handelt es sich bei diesen Traces um Teil-Traces, nicht um vollständige Transaktionen. end-to-end

Sobald das ADOT SDK eine Spanne mit Anomalien erkennt, versucht es, so viele Spans wie möglich aus derselben Trace auszusenden. Alle im Rahmen dieser Funktion ausgegebenen Spans sind mit dem Attribut, gekennzeichnet. `aws.trace.flag.sampled = 0` Auf diese Weise können Sie bei der Transaktionssuche und -analyse auf einfache Weise partielle Spuren (Erfassung von Anomalien) von vollständigen Spuren (normale Probenahme) unterscheiden.

Wir empfehlen, Transaction Search zu integrieren, um partielle Traces anzeigen und abfragen zu können. Das folgende Beispiel zeigt eine Serviceseite in der Application Signals Console. ServiceC ist mit der Erfassung von Anomalie-Spannen konfiguriert und ist Teil einer Anrufkette, bei der der Sampling-Boost angewendet wird. Diese Konfiguration generiert sowohl vollständige als auch teilweise Traces. Sie können das `aws.trace.flag.sampled` Attribut verwenden, um zwischen Trace-Typen zu unterscheiden.



Die Erfassung von Anomaliespannen kann nur über den aktiviert oder angepasst werden. [Lokale SDK-Konfiguration]

Lokale SDK-Konfiguration

Sie können Funktionen für adaptives Sampling im ADOT SDK konfigurieren, indem Sie eine YAML-Konfiguration über eine Umgebungsvariable bereitstellen. Die lokale Konfiguration bietet eine detaillierte Kontrolle über Anomaliebedingungen und Schwellenwerte.

Dies ist für die Erfassung von Anomaliebereichen und optional für die Anpassung der Bedingungen für den Probenahmeanstieg erforderlich. Im Folgenden finden Sie ein Beispiel für die Konfiguration:

```
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
```

Die Felddefinitionen sind unten aufgeführt:

- `version`— Schemaversion für die Konfigurationsdatei
- `anomalyConditions`— Definiert die Bedingungen, unter denen Anomalien erkannt werden, und wie sie verwendet werden
  - `errorCodeRegex`— Regulärer Ausdruck, der definiert, welche HTTP-Statuscodes als Anomalien betrachtet werden
  - `operations`— Liste der Operationen oder Endpunkte, für die die Bedingung gilt
  - `highLatencyMs`— Latenzschwellenwert (in Millisekunden), bei dessen Überschreitung Zeitspannen als Anomalien behandelt werden
  - `usage`— Definiert, für welches Feature die Bedingung gilt:
    - `both`— Gilt für Sampling-Boost und Erfassung von Anomalie-Spannen (Standardeinstellung, wenn keine Verwendung angegeben ist)
    - `sampling-boost`— Wird nur zum Auslösen von Sampling-Boosts verwendet
    - `anomaly-span-capture`— Wird nur für die Erfassung von Anomaliebereichen verwendet
- `anomalyCaptureLimit`— Definiert Grenzwerte dafür, wie viele Spuren mit Anomaliespannen ausgegeben werden.

  `anomalyTracesPerSecond`— Maximale Anzahl von Traces, die pro Sekunde erfasst werden, um eine übermäßige Spanne zu verhindern (der Standardwert ist 1, falls nicht anomalyCaptureLimit vorhanden).

> **Note**
>
> - `AnomalyConditions`überschreibt die standardmäßige Anomaliebedingung für Sampling-Boost (HTTP 5xx). Wenn Sie die Standardbedingung beibehalten möchten, während Sie die lokale Konfiguration verwenden, müssen Sie sie explizit in jedes Element von aufnehmen. `AnomalyConditions`
> - Für jedes `anomalyConditions` Element:

- Wenn das `operations` Feld weggelassen wird, gilt die Bedingung für alle Operationen (Service Level)
- Wenn das `operations` Feld zwar vorhanden, aber auf eine leere Liste gesetzt ist, gilt die Bedingung für „Keine Operationen“, sodass dieses Element nicht aktiviert ist
- Wenn `errorCodeRegex` sowohl als auch weggelassen `highLatencyMs` werden, gibt es für die Bedingung keine zu bewertenden Anomaliekriterien, sodass dieses Element nicht berücksichtigt werden kann

- Logische Beziehungen:
  - Zwischen den Elementen in `anomalyConditions` der Datei ist die Beziehung ODER.
  - Innerhalb eines einzelnen Elements werden mehrere Felder (z. B. `errorCodeRegex` und`highLatencyMs`) mit UND kombiniert.

    Zum Beispiel:

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

    Diese Bedingung bedeutet, dass der Statuscode 429 oder 5xx UND einer Latenz von ≥ 300 ms entspricht.

Wenden Sie die lokale Konfiguration auf das ADOT SDK an

Sie können die lokale Konfiguration auf das ADOT SDK anwenden, indem Sie die Umgebungsvariable festlegen. `AWS_XRAY_ADAPTIVE_SAMPLING_CONFIG` Der Wert muss ein gültiges YAML-Dokument sein (inline oder verschachtelt).

Beispielsweise legen Amazon EC2 und Amazon ECS die Umgebungsvariable direkt fest:

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

Definieren Sie für Amazon EKS die Umgebungsvariable in der Pod-Spezifikation als verschachteltes YAML:

```
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
```

Deep Linking für Konsolen

Sie können Routen und Abfragen verwenden, um Deeplinks zu bestimmten Traces oder gefilterten Ansichten von Traces und der Trace-Map zu erstellen.

Konsolenseiten

- Willkommensseite — xray/home#/welcome
- Erste Schritte — xray/home#/getting-gestartet
- Trace-Map — xray/home#/service -map
- Spuren — xray/home#/traces

Ablaufverfolgungen

Sie können Links für Timeline-, Roh- und Kartenansichten einzelner Ablaufverfolgungen generieren.

Zeitleiste verfolgen — `xray/home#/traces/`*`trace-id`*

Unverarbeitete Trace-Daten — `xray/home#/traces/`*`trace-id`*`/raw`

Example— rohe Trace-Daten

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

Filterausdrücke

Link zu einer gefilterten Liste von Ablaufverfolgungsdaten.

Ansicht gefilterter Spuren — `xray/home#/traces?filter=`*`filter-expression`*

Example— Ausdruck filtern

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

Example— Filterausdruck (URL-kodiert)

```
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
```

Weitere Informationen zu Filterausdrücken finden Sie unter Verwenden von Filterausdrücken.

Zeitraum

Geben Sie eine Zeitdauer oder Start- und Endzeit im Format ISO86 01 an. Zeitbereiche sind in UTC angegeben und können bis zu 6 Stunden lang sein.

Länge der Zeit — `xray/home#/`*`page`*`?timeRange=`*`range-in-minutes`*

Example— Trace-Map für die letzte Stunde

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

Start- und Endzeit — `xray/home#/`*`page`*`?timeRange=`*`start`*`~`*`end`*

Example— Sekundengenauer Zeitbereich

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

Example— minutengenauer Zeitbereich

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

Region

Geben Sie AWS-Region an, ob auf Seiten in dieser Region verlinkt werden soll. Wenn Sie keine Region angeben, führt Sie die Konsole zur zuletzt besuchten Region.

Region — `xray/home`**`?region=`*`region`***`#/`*`page`*

Example— Streckenkarte in den USA West (Oregon) (us-west-2)

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

Wenn Sie eine Region mit anderen Abfrageparametern angeben, steht die Regionsabfrage vor dem Hash und die X-Ray-specific Abfragen hinter dem Seitennamen.

Example— Streckenkarte für die letzte Stunde in US West (Oregon) (us-west-2)

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

Kombiniert

Example— aktuelle Spuren mit einem Dauerfilter

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

Output

- Seite — Spuren
- Zeitraum — Letzte 15 Minuten
- Filter — Dauer >= 5 UND Dauer <= 8

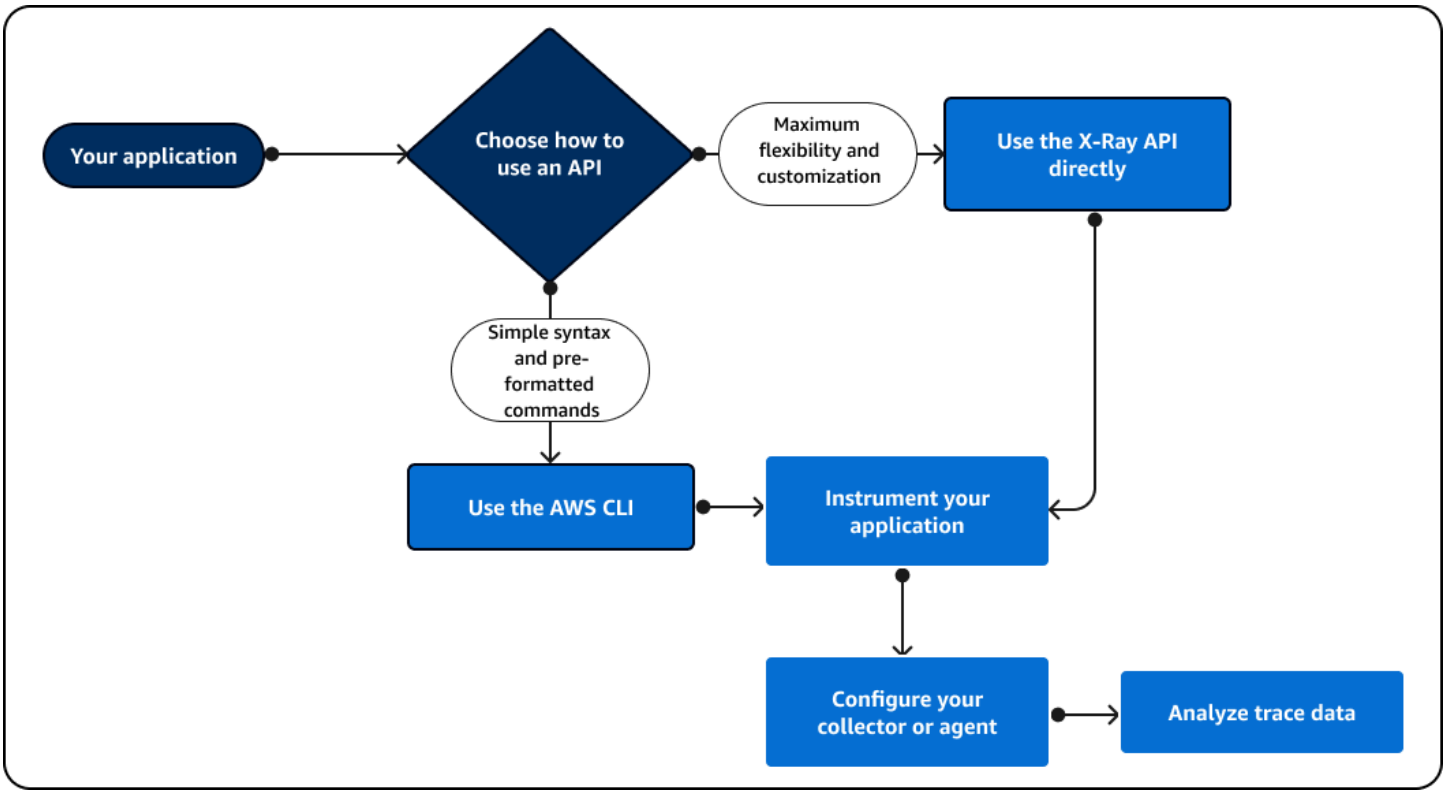
Verwenden Sie die X-Ray-API

Wenn das X-Ray-SDK Ihre Programmiersprache nicht unterstützt, können Sie entweder X-Ray APIs direkt oder AWS Command Line Interface (AWS CLI) verwenden, um X-Ray-API-Befehle aufzurufen. Wählen Sie anhand der folgenden Anleitung aus, wie Sie mit der API interagieren möchten:

- Verwenden Sie die AWS CLI einfachere Syntax mit vorformatierten Befehlen oder mit Optionen in Ihrer Anfrage.
- Verwenden Sie die X-Ray-API direkt für maximale Flexibilität und Anpassung bei Anfragen, die Sie an X-Ray stellen.

Wenn Sie die X-Ray-API direkt anstelle von verwenden AWS CLI, müssen Sie Ihre Anfrage im richtigen Datenformat parametrisieren und möglicherweise auch die Authentifizierung und Fehlerbehandlung konfigurieren.

Das folgende Diagramm zeigt eine Anleitung zur Auswahl der Interaktion mit der X-Ray-API:



Verwenden Sie die X-Ray-API, um Trace-Daten direkt an X-Ray zu senden. Die X-Ray-API unterstützt alle Funktionen, die im X-Ray SDK verfügbar sind, einschließlich der folgenden allgemeinen Aktionen:

- PutTraceSegments— Lädt Segmentdokumente auf X-Ray hoch.
- BatchGetTraces— Ruft eine Liste von Traces in einer Trace-Liste ab. IDs Jeder abgerufene Trace ist eine Sammlung von Segmentdokumenten aus einer einzigen Anfrage.
- GetTraceSummaries— Ruft IDs Traces ab und kommentiert sie. Sie können a angeben`FilterExpression`, um eine Teilmenge der Trace-Zusammenfassungen abzurufen.
- GetTraceGraph— Ruft ein Service-Diagramm für eine bestimmte Trace-ID ab.
- GetServiceGraph— Ruft eine ab JSON formatiertes Dokument, das Dienste beschreibt, die eingehende Anfragen verarbeiten und nachgelagerte Anfragen aufrufen.

Sie können auch das AWS Command Line Interface (AWS CLI) in Ihrem Anwendungscode verwenden, um programmgesteuert mit X-Ray zu interagieren. Das AWS CLI unterstützt alle Funktionen, die im X-Ray SDK verfügbar sind, einschließlich der Funktionen für andere AWS-Services. Bei den folgenden Funktionen handelt es sich um Versionen der zuvor aufgeführten API-Operationen mit einem einfacheren Format:

- put-trace-segments— Lädt Segmentdokumente auf X-Ray hoch.
- batch-get-traces— Ruft eine Liste von Traces in einer Trace-Liste ab. IDs Jeder abgerufene Trace ist eine Sammlung von Segmentdokumenten aus einer einzigen Anfrage.
- get-trace-summaries— Ruft IDs Traces ab und kommentiert sie. Sie können a angeben`FilterExpression`, um eine Teilmenge der Trace-Zusammenfassungen abzurufen.
- get-trace-graph— Ruft ein Service-Diagramm für eine bestimmte Trace-ID ab.
- get-service-graph— Ruft ein `JSON` formatiertes Dokument ab, das Dienste beschreibt, die eingehende Anfragen verarbeiten und nachgelagerte Anfragen aufrufen.

Um zu beginnen, müssen Sie das AWS CLIfür Ihr Betriebssystem installieren. AWS unterstützt Linux, macOS and Windows Betriebssysteme. Weitere Informationen zur Liste der X-Ray-Befehle finden Sie in der AWS CLI Befehlsreferenz für X-Ray.

Themen

- Verwenden der AWS X-Ray API mit der AWS CLI
- Senden von Ablaufverfolgungsdaten an AWS X-Ray
- Daten abrufen von AWS X-Ray
- Konfigurieren von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray -API

- Verwenden von Sampling-Regeln mit der X-Ray-API
- AWS X-Ray Dokumente segmentieren

Verwenden der AWS X-Ray API mit der AWS CLI

Über die AWS CLI können Sie direkt auf den X-Ray-Dienst zugreifen und denselben Dienst verwenden APIs , den die X-Ray-Konsole zum Abrufen des Service-Graphen und der Trace-Rohdaten verwendet. Die Beispielanwendung enthält Skripts, die zeigen, wie diese APIs mit der AWS CLI verwendet werden.

Voraussetzungen

In diesem Tutorial werden die Scorekeep-Beispielanwendung und darin enthaltene Skripts verwendet, um Ablaufverfolgungsdaten und eine Service-Übersicht zu erstellen. Folgen Sie den Anweisungen im Tutorial "Erste Schritte", um die Anwendung zu starten.

In diesem Tutorial wird AWS CLI die grundlegende Verwendung der X-Ray-API veranschaulicht. Die AWS CLI, verfügbar für Windows, Linux und OS-X, bietet öffentlichen Befehlszeilenzugriff APIs für alle AWS-Services.

> **Note**
>
> Sie müssen überprüfen, ob Ihr System für dieselbe Region konfiguriert AWS CLI ist, in der Ihre Scorekeep-Beispielanwendung erstellt wurde.

Die enthaltenen Skripts zum Testen der Beispielanwendung verwenden `cURL`, um Datenverkehr zur API zu senden, und `jq` zum Analysieren der Ausgabe. Sie können die ausführbare `jq`-Datei von stedolan.github.io und die ausführbare `curl`-Datei von https://curl.haxx.se/download.html herunterladen. Die meisten Linux- und OS X-Installationen umfassen cURL.

Generieren von Ablaufverfolgungsdaten

Die Web-App generiert weiterhin alle paar Sekunden Datenverkehr zur API, während das Spiel fortgesetzt wird, es wird aber nur eine Art der Anforderung generiert. Verwenden Sie das Skript `test-api.sh`, um End-to-End-Szenarien auszuführen und um während der Prüfung der API vielfältigere Ablaufverfolgungsdaten zu generieren.

So verwenden Sie das **`test-api.sh`**-Skript

1. In der Elastic-Beanstalk-Konsole öffnen.
2. Navigieren Sie zur Managementkonsole für Ihre Umgebung.
3. Kopieren Sie die Umgebungs-URL aus dem Header der Seite.
4. Öffnen Sie `bin/test-api.sh` und ersetzen Sie den Wert für die API mit der URL Ihrer Umgebung.

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

5. Führen Sie das Skript aus, um Datenverkehr zur API zu generieren.

```
~/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
```

Verwenden Sie die X-Ray-API

Die AWS CLI stellt Befehle für alle API-Aktionen bereit, die X-Ray bereitstellt, einschließlich `GetServiceGraph`und `GetTraceSummaries`. Weitere Informationen über alle unterstützten Aktionen und die von ihnen verwendeten Datentypen finden Sie in der AWS X-Ray -API-Referenz.

Example bin/service-graph.sh

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

Das Skript ruft ein Service-Diagramm für die letzten 10 Minuten ab.

```
~/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'
```

Das Skript ruft die URLs Traces ab, die vor einer bis zwei Minuten generiert wurden.

```
~/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[*]'
```

Das Skript ruft die vollständigen Ablaufverfolgungsdaten ab, die ein bis zwei Minuten zuvor generiert wurden.

```
~/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
    }
...
```


Bereinigen

Beenden Sie Ihre Elastic Beanstalk Beanstalk-Umgebung, um die EC2 Amazon-Instances, DynamoDB-Tabellen und andere Ressourcen herunterzufahren.

So beenden Sie die Elastic Beanstalk-Umgebung

1. In der Elastic-Beanstalk-Konsole öffnen.
2. Navigieren Sie zur Managementkonsole für Ihre Umgebung.
3. Wählen Sie Aktionen.
4. Wählen Sie Terminate Environment.
5. Wähen Sie Beenden.

Trace-Daten werden nach 30 Tagen automatisch aus X-Ray gelöscht.

Senden von Ablaufverfolgungsdaten an AWS X-Ray

Sie können Trace-Daten in Form von Segmentdokumenten an X-Ray senden. Ein Segmentdokument ist eine JSON-formatierte Zeichenfolge mit Informationen über die Arbeit, die Ihre Anwendung im Dienste einer Anforderung verrichtet. Ihre Anwendung kann Daten über die Arbeit, die sie selbst in Segmenten verrichtet, oder über Arbeit, die Downstream-Services und -Ressourcen in Untersegmenten verrichtet, aufzeichnen.

Segmente zeichnen Informationen über die Arbeit auf, die Ihre Anwendung verrichtet. Ein Segment zeichnet mindestens die Zeit auf, die für eine Aufgabe aufgewendet wurde, einen Namen und zwei IDs. Die Nachverfolgungs-ID zeichnet die Anforderung auf Ihrem Weg zwischen den Services auf. Die Segment-ID verfolgt die für die Anforderung durch einen einzelnen Service verrichtete Arbeit.

Example Minimales vollständiges Segment

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

Wenn eine Anfrage empfangen wird, können Sie eine In-Progress-Segment als Platzhalter senden, bis die Anforderung abgeschlossen ist.

Example In Bearbeitung befindliches Segment

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

Sie können Segmente direkt, mit oder über den `PutTraceSegments`X-Ray-Daemon an X-Ray senden.

Die meisten Anwendungen rufen mit dem AWS SDK andere Dienste auf oder greifen auf Ressourcen zu. Zeichnen Sie Informationen über Downstream-Aufrufe in Untersegmenten auf. X-Ray verwendet Untersegmente, um Downstream-Services zu identifizieren, die keine Segmente senden, und Einträge für sie im Service-Graph zu erstellen.

Ein Untersegment kann in einem vollständigen Segmentdokument eingebettet oder separat gesendet werden. Senden Sie Untersegmente separat, um Downstream-Aufrufe für lang andauernde Anfragen asynchron zu verfolgen oder um zu verhindern, dass die maximale Segmentdokumentgröße (64 kB) überschritten wird.

Example Untersegment

Ein Untersegment hat den `type` `subsegment` und eine `parent_id`, mit der das übergeordnete Segment identifiziert wird.

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

Weitere Informationen über die Felder und Werte, die in Segmente und Untersegmente eingegeben werden können, finden Sie unter AWS X-Ray Dokumente segmentieren.

Sections

- Trace wird generiert IDs
- Benutzen PutTraceSegments
- Segmentdokumente an den X-Ray-Daemon senden

Trace wird generiert IDs

Um Daten an X-Ray zu senden, müssen Sie für jede Anfrage eine eindeutige Trace-ID generieren.

Röntgen-Trace-ID-Format

Ein X-Ray `trace_id` besteht aus drei Zahlen, die durch Bindestriche getrennt sind. Beispiel, `1-58406520-a006649127e371903a2de979`. Dies umfasst:

- Die Versionsnummer, die ist. 1
- Die Uhrzeit der ursprünglichen Anfrage in Unix-Epochenzeit unter Verwendung von 8 Hexadezimalziffern.

  Zum Beispiel ist 10:00 Uhr am 1. Dezember 2016 PST in Epochenzeit `1480615200` Sekunden oder `58406520` in Hexadezimalziffern.
- Eine weltweit eindeutige 96-Bit-ID für den Trace mit 24 Hexadezimalziffern.

> **Note**
>
> X-Ray unterstützt jetzt IDs Traces, die mit OpenTelemetry und jedem anderen Framework erstellt wurden, das der W3C Trace Context-Spezifikation entspricht. Eine W3C-Trace-ID muss beim Senden an X-Ray im X-Ray-Trace-ID-Format formatiert werden. Beispielsweise `4efaaf4d1e8720b39541901950019ee5` sollte die W3C-Trace-ID wie `1-4efaaf4d-1e8720b39541901950019ee5` beim Senden an X-Ray formatiert werden. X-Ray-Trace IDs enthält den ursprünglichen Anforderungszeitstempel in der Unix-Epochenzeit, aber dies ist nicht erforderlich, wenn W3C-Trace IDs im X-Ray-Format gesendet wird.

Sie können ein Skript schreiben, um X-Ray Trace IDs für Tests zu generieren. Nachfolgend finden Sie zwei Beispiele.

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"
```

In der Scorekeep-Beispielanwendung finden Sie Skripte, die Trace erstellen IDs und Segmente an den X-Ray-Daemon senden.

- Python – `xray_start.py`
- Bash — `xray_start.sh`

Benutzen PutTraceSegments

Sie können Segmentdokumente mit der `PutTraceSegments`-API hochladen. Die API hat einen einzelnen Parameter, `TraceSegmentDocuments`, der eine Liste von JSON-Segmentdokumenten aufnimmt.

Verwenden Sie mit der AWS-CLI den `aws xray put-trace-segments` Befehl, um Segmentdokumente direkt an X-Ray zu senden.

```
$ 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**
>
> Windows Command Processor und Windows PowerShell haben unterschiedliche Anforderungen für das Anführungszeichen und das Maskieren von Anführungszeichen in JSON-Zeichenketten. Weitere Details finden Sie unter Setzen von Anführungszeichen für Zeichenfolgen im AWS CLI -Benutzerhandbuch.

Die Ausgabe führt alle Segmente auf, deren Verarbeitung fehlgeschlagen ist. Beispiel: Wenn das Datum in der Nachverfolgungs-ID zu weit in der Vergangenheit liegt, sehen Sie eine Fehlermeldung wie die folgende.

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

Sie können mehrere Segmentdokumente gleichzeitig, durch Leerräume getrennt, weitergeben.

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

Segmentdokumente an den X-Ray-Daemon senden

Anstatt Segmentdokumente an die X-Ray-API zu senden, können Sie Segmente und Untersegmente an den X-Ray-Daemon senden, der sie zwischenspeichert und stapelweise auf die X-Ray-API hochlädt. Das X-Ray-SDK sendet Segmentdokumente an den Daemon, um AWS direkte Aufrufe zu vermeiden.

> **Note**
>
> Informationen zur Ausführung des Daemons finden Sie unter Lokales Ausführen des X-Ray-Daemons.

Senden Sie das Segment in JSON über den UDP-Port 2000 mit dem vorangestellten Daemon-Kopf `{"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"}
```

Unter Linux können Sie Segmentdokumente von einem Bash-Terminal aus an den Daemon senden. Speichern Sie den Kopf und das Segmentdokument in einer Textdatei, und senden Sie sie an `/dev/udp` mit `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"}
```

Überprüfen Sie das Daemon-Protokoll, um sicherzustellen, dass das Segment an X-Ray gesendet wurde.

```
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)
```

Daten abrufen von AWS X-Ray

AWS X-Ray verarbeitet die Trace-Daten, die Sie an sie senden, um vollständige Traces, Trace-Zusammenfassungen und Service-Diagramme in JSON zu generieren. Sie können die generierten Daten mit der AWS CLI direkt von der API abrufen.

Sections

Abrufen des Service-Diagramms

Sie können die API `GetServiceGraph` verwenden, um das JSON-Service-Diagramm abzurufen. Die API erfordert eine Start- und Endzeit. Sie können diese mit dem `date`-Befehl an einem Linux-Terminal berechnen.

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

`date +%s` druckt ein Datum in Sekunden. Verwenden Sie diese Zahl als Endzeit. Ziehen Sie Zeit ab, um eine Startzeit zu erhalten.

Example Script, um ein Service-Diagramm für die letzten 10 Minuten abzurufen

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

Das folgende Beispiel zeigt ein Service-Diagramm mit 4 Knoten, darunter einen Client-Knoten, eine EC2 Instance, eine DynamoDB-Tabelle und ein Amazon SNS SNS-Thema.

Example GetServiceGraph Ausgabe

```
{
    "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
```

```
                }
            ]
        }
    ]
}
```

Abrufen des Service-Diagramms nach Gruppe

Zum Abrufen eines Service-Diagramms basierend auf dem Inhalt einer Gruppe geben Sie einen `groupName` oder `groupARN` an. Das folgende Beispiel zeigt einen Service-Diagrammaufruf einer Gruppe mit dem Namen Example1.

Example Skript zum Abrufen eines Service-Diagramms nach Name für die Gruppe Example1

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

Abrufen von Ablaufverfolgungen

Sie können die API `GetTraceSummaries` verwenden, um eine Liste von Ablaufverfolgungsübersichten abzurufen. Trace-Zusammenfassungen enthalten Informationen, anhand derer Sie Traces identifizieren können, die Sie vollständig herunterladen möchten, einschließlich Anmerkungen, Anfrage- und Antwortinformationen und. IDs

Beim Aufruf von `aws xray get-trace-summaries` stehen zwei `TimeRangeType`- Flags zur Verfügung:

- TraceId— Die `GetTraceSummaries` Standardsuche verwendet die TraceID-Zeit und gibt Traces zurück, die innerhalb des berechneten `[start_time, end_time)` Bereichs gestartet wurden. Dieser Bereich von Zeitstempeln wird auf der Grundlage der Kodierung des Zeitstempels innerhalb von berechnet oder kann TraceId manuell definiert werden.
- Ereigniszeit — Um nach Ereignissen zu suchen, die sich im Laufe der Zeit ereignen, ermöglicht AWS X-Ray die Suche nach Spuren mithilfe von Ereigniszeitstempeln. Die Ereigniszeit gibt Traces zurück, die während des `[start_time, end_time)`-Bereichs aktiv sind, unabhängig davon, wann der Trace begonnen hat.

Mit dem Befehl `aws xray get-trace-summaries` können Sie eine Liste von Ablaufverfolgungsübersichten abrufen. Die folgenden Befehle rufen unter Verwendung der TraceId Standardzeit eine Liste von Trace-Zusammenfassungen ab, die zwischen 1 und 2 Minuten zurückliegen.

Example Skript zum Abrufen von Ablaufverfolgungsübersichten

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

Example GetTraceSummaries Ausgabe

```
{
    "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
}
```

Mithilfe der Ablaufverfolgungs-ID aus der Ausgabe können Sie mit der API `BatchGetTraces` eine vollständige Ablaufverfolgung abrufen.

Example BatchGetTraces Befehl

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

Example BatchGetTraces Ausgabe

```
{
    "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": []
}
```


Die vollständige Ablaufverfolgung umfasst ein Dokument für jedes Segment, das aus allen Segmentdokumenten kompiliert wurde, die mit der gleichen Ablaufverfolgungs-ID erhalten wurden. Diese Dokumente stellen nicht die Daten dar, wie sie durch Ihre Anwendung an X-Ray gesendet wurden. Stattdessen stellen sie die verarbeiteten Dokumente dar, die vom X-Ray-Dienst generiert wurden. X-Ray erstellt das vollständige Trace-Dokument, indem es die von

Ihrer Anwendung gesendeten Segmentdokumente kompiliert und Daten entfernt, die nicht dem Segmentdokumentschema entsprechen.

X-Ray erstellt auch abgeleitete Segmente für Downstream-Aufrufe an Dienste, die selbst keine Segmente senden. Wenn Sie DynamoDB beispielsweise mit einem instrumentierten Client aufrufen, zeichnet das X-Ray-SDK ein Untersegment mit Details zum Aufruf aus seiner Sicht auf. DynamoDB sendet jedoch kein entsprechendes Segment. X-Ray verwendet die Informationen im Untersegment, um ein abgeleitetes Segment zu erstellen, das die DynamoDB-Ressource in der Trace-Map darstellt, und fügt es dem Trace-Dokument hinzu.

Um mehrere Traces von der API abzurufen, benötigen Sie eine Trace-Liste IDs, die Sie mit einer Abfrage aus der `get-trace-summaries` Ausgabe extrahieren können.AWS CLI Leiten Sie die Liste zur Eingabe von `batch-get-traces` um, um vollständige Traces für einen bestimmten Zeitraum zu erhalten.

Example Script, um vollständige Ablaufverfolgungen für den Zeitraum von einer Minute zu erhalten

```
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[*]'
```

Abrufen und Anpassen der Ursachenanalyse

Nach der Generierung einer Trace-Zusammenfassung mit der GetTraceSummaries API können unvollständige Trace-Zusammenfassungen in ihrem JSON-Format wiederverwendet werden, um einen verfeinerten Filterausdruck zu erstellen, der auf den Grundursachen basiert. In den nachstehenden Beispielen finden Sie eine exemplarische Vorgehensweise für die Optimierungsschritte.

Example GetTraceSummaries Beispielausgabe — Abschnitt zur Hauptursache bei der Antwortzeit

```
{
  "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
        }
      ]
    }
  ]
}
```

Durch das Bearbeiten und Vornehmen von Auslassungen der oben genannten Ausgaben kann dieses JSON-Objekt ein Filter für übereinstimmende Ursachenentitäten werden. Für jedes Feld im JSON-Objekt müssen alle Bewerberabgleiche exakt erfolgen. Andernfalls wird die Nachverfolgung nicht zurückgegeben. Entfernte Felder werden zu Platzhalterwerten, ein Format, das mit der Abfragestruktur des Filterausdrucks kompatibel ist.

Example Reaktionszeit „Ursache“ neu formatiert

```
{
  "Services": [
    {
      "Name": "GetWeatherData",
      "EntityPath": [
```

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

Dieses JSON-Objekt wird dann als Teil eines Filterausdrucks durch einen Aufruf von `rootcause.json = #[{}]` verwendet. Weitere Informationen zu Abfragen mit Filterausdrücken finden Sie im Kapitel Filterausdrücke.

Example Beispiel für JSON-Filter

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

Konfigurieren von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray -API

AWS X-Ray ermöglicht APIs die Konfiguration von Sampling-Regeln, Gruppenregeln und Verschlüsselungseinstellungen.

Sections

- Verschlüsselungseinstellungen
- Samplingregeln
- Gruppen

Verschlüsselungseinstellungen

Dient PutEncryptionConfigzur Angabe eines AWS Key Management Service (AWS KMS) - Schlüssels, der für die Verschlüsselung verwendet werden soll.

> **Note**
>
> X-Ray unterstützt keine asymmetrischen KMS-Schlüssel.

```
$ 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"
    }
}
```

Für die Schlüssel-ID können Sie einen Alias verwenden (wie im Beispiel gezeigt), eine Schlüssel-ID oder einen Amazon-Ressourcennamen (ARN).

Verwenden Sie GetEncryptionConfig, um die aktuelle Konfiguration abzurufen. Wenn X-Ray die Übernahme Ihrer Einstellungen abgeschlossen hat, ändert sich der Status von `UPDATING` zu`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"
    }
}
```

Um die Verwendung eines KMS-Schlüssels zu beenden und die Standardverschlüsselung zu verwenden, setzen Sie den Verschlüsselungstyp auf`NONE`.

```
$ aws xray put-encryption-config --type NONE
{
```

```
    "EncryptionConfig": {
        "Status": "UPDATING",
        "Type": "NONE"
    }
}
```

Samplingregeln

Sie können die Sampling-Regeln in Ihrem Konto mit der X-Ray-API verwalten. Weitere Informationen zum Hinzufügen und Verwalten von Tags finden Sie unterKennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

Sie können alle Samplingregeln mit `GetSamplingRules` abrufen.

```
$ 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
        }
    ]
}
```

Die Standardregel gilt für alle Anfragen, die nicht mit einer anderen Regel übereinstimmen. Es ist die Regel mit der niedrigsten Priorität, die nicht gelöscht werden kann. Sie können jedoch die Rate und die Reservoirgröße mit `UpdateSamplingRule` ändern.

Example API-Eingabe für **UpdateSamplingRule**— 10000-default.json

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

Das folgende Beispiel verwendet die vorherige Datei als Eingabe, um die Standardregel in ein Prozent ohne Reservoir zu ändern. Tags sind optional. Wenn Sie Tags hinzufügen möchten, ist ein Tag-Schlüssel erforderlich, und Tag-Werte sind optional. Um vorhandene Tags aus einer Stichprobenregel zu entfernen, verwenden Sie 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
        },
```

Erstellen Sie zusätzliche Samplingregeln mit `CreateSamplingRule`. Wenn Sie eine Regel erstellen, sind die meisten Regelfelder erforderlich. Das folgende Beispiel erstellt zwei Regeln. Diese erste Regel legt einen Basissatz für die Scorekeep-Beispielanwendung fest. Er entspricht

allen Anforderungen, die von der API bereitgestellt werden, die nicht mit einer höheren Priorität übereinstimmen.

Example API-Eingabe für **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
    }
}
```

Die zweite Regel gilt auch für Scorekeep, hat aber eine höhere Priorität und ist spezifischer. Diese Regel enthält eine sehr niedrige Samplingrate für Polling-Anfragen. Dies sind GET-Anfragen, die der Client alle paar Sekunden sendet, um eine Prüfung auf Änderungen am Spielstatus durchzuführen.

Example API-Eingabe für **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
    }
}
```

Tags sind optional. Wenn Sie Tags hinzufügen möchten, ist ein Tag-Schlüssel erforderlich, und Tag-Werte sind optional.

```
$ 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
    }
}
```

Um eine Samplingregel zu löschen, verwenden Sie [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
    }
}
```

Gruppen

Sie können die X-Ray-API verwenden, um Gruppen in Ihrem Konto zu verwalten. Gruppen sind eine Sammlung von Ablaufverfolgungen, die durch einen Filterausdruck definiert sind. Sie können Gruppen verwenden, um zusätzliche Servicegraphen zu erstellen und CloudWatch Amazon-Metriken bereitzustellen. Weitere Informationen Daten abrufen von AWS X-Ray zur Arbeit mit Servicediagrammen und Metriken über die X-Ray-API finden Sie unter. Weitere Informationen zu Gruppen finden Sie unterGruppen konfigurieren. Weitere Informationen zum Hinzufügen

und Verwalten von Stichwörtern finden Sie unterKennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

Erstellen Sie eine Gruppe mit `CreateGroup`. Tags sind optional. Wenn Sie Tags hinzufügen möchten, ist ein Tag-Schlüssel erforderlich, und Tag-Werte sind optional.

```
$ 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}"
}
```

Rufen Sie alle vorhandenen Gruppen mit `GetGroups` ab.

```
$ 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"
}
```

Aktualisieren Sie eine Gruppe mit `UpdateGroup`. Tags sind optional. Wenn Sie Tags hinzufügen möchten, ist ein Tag-Schlüssel erforderlich, und Tag-Werte sind optional. Um vorhandene Tags aus einer Gruppe zu entfernen, verwenden Sie 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}"
}
```

Löschen Sie eine Gruppe mit `DeleteGroup`.

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

Verwenden von Sampling-Regeln mit der X-Ray-API

Das AWS X-Ray SDK verwendet die X-Ray-API, um Probenahmeregeln abzurufen, Probenahmeergebnisse zu melden und Kontingente abzurufen. Sie können diese verwenden APIs , um besser zu verstehen, wie Sampling-Regeln funktionieren, oder um Sampling in einer Sprache zu implementieren, die das X-Ray SDK nicht unterstützt.

Rufen Sie zunächst alle Samplingregeln mit `GetSamplingRules` ab.

```
$ 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
        }
    ]
}
```

Die Ausgabe enthält die Standardregel und benutzerdefinierte Regeln. Weitere Informationen finden Sie unter Samplingregeln, wenn Sie noch keine Samplingregeln erstellt haben.

Bewerten Sie Regeln anhand eingehender Anfragen in aufsteigender Reihenfolge der Priorität. Wenn eine Regel übereinstimmt, verwenden Sie die feste Rate und eine feste Reservoirgröße, um eine Samplingentscheidung zu treffen. Zeichnen Sie die per Stichprobe geprüften Anforderungen auf und ignorieren Sie (für die Nachverfolgung) nicht geprüfte Anfragen. Beenden Sie die Bewertung der Regeln, wenn eine Samplingentscheidung getroffen wird.

Eine Regel-Reservoirgröße ist die Zielanzahl der Ablaufverfolgungen pro Sekunde, bevor Sie die feste Rate anwenden. Das Reservoir gilt kumulativ für alle Services, sodass Sie es nicht direkt verwenden können. Wenn sie jedoch ungleich Null ist, können Sie eine Spur pro Sekunde aus dem Reservoir ausleihen, bis X-Ray eine Quote zuweist. Vor dem Empfang eines Kontingents zeichnen Sie die erste Anfrage pro Sekunde auf und wenden die feste Rate auf zusätzliche Anfragen an. Die feste Rate ist ein Dezimalwert zwischen 0 und 1,00 (100 %).

Das folgende Beispiel zeigt einen Aufruf von `GetSamplingTargets` mit Details zu Samplingentscheidungen in den letzten 10 Sekunden.

```
$ 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": []
}
```

Die Antwort von X-Ray beinhaltet ein Kontingent, das verwendet werden kann, anstatt Kredite aus dem Reservoir aufzunehmen. In diesem Beispiel hat sich der Service aus dem Reservoir 10 Ablaufverfolgungen über 10 Sekunden geliehen und die feste Rate von 10 % auf den anderen 100 Anfragen angewendet. Daraus ergeben sich 20 geprüfte Anfragen. Das Kontingent ist für fünf Minuten gültig (angegeben durch die Gültigkeitsdauer) oder bis ein neues Kontingent zugewiesen wird. X-Ray kann auch ein längeres Berichtsintervall als das Standardintervall zuweisen, obwohl dies hier nicht der Fall war.

> **Note**
>
> Die Antwort von X-Ray enthält möglicherweise kein Kontingent, wenn Sie sie zum ersten Mal aufrufen. Leihen Sie weiterhin vom Reservoir, bis Sie ein Kontingent zugewiesen bekommen.

Die anderen zwei Felder in der Antwort können möglicherweise Konflikte mit der Eingabe anzeigen. Vergleichen Sie `LastRuleModification` mit dem letzten Aufruf von `GetSamplingRules`. Wenn er neuer ist, rufen Sie eine Kopie der Regeln ab. `UnprocessedStatistics` kann Fehler enthalten, die angeben, dass eine Regel gelöscht wurde, dass das Statistikdokument in der Eingabe zu alt war oder Berechtigungsfehler aufgetreten sind.

AWS X-Ray Dokumente segmentieren

Ein Ablaufverfolgungssegment ist eine JSON-Darstellung einer Anfrage, die Ihrer Anwendung dient. Ein Trace-Segment zeichnet Informationen über die ursprüngliche Anfrage, Informationen über die Arbeit auf lokaler Ebene und Untersegmente mit Informationen über Downstream-Aufrufe auf, die Ihre Anwendung an AWS Ressourcen APIs, HTTP- und SQL-Datenbanken durchführt.

Ein Segmentdokument übermittelt Informationen über ein Segment an X-Ray. Ein Segmentdokument kann bis zu 64 kB groß sein und ein ganzes Segment mit Untersegmenten, ein Fragment eines Segments, das angibt, dass eine Anfrage bearbeitet wird, oder ein einzelnes Untersegment, das separat gesendet wird, enthalten. Mithilfe der `PutTraceSegments`API können Sie Segmentdokumente direkt an X-Ray senden.

X-Ray kompiliert und verarbeitet Segmentdokumente, um abfragbare Trace-Zusammenfassungen und vollständige Traces zu generieren, auf die Sie jeweils mit und zugreifen können. `GetTraceSummariesBatchGetTraces` APIs Zusätzlich zu den Segmenten und Untersegmenten, die Sie an X-Ray senden, verwendet der Service Informationen in Untersegmenten, um abgeleitete Segmente zu generieren, und fügt sie dem vollständigen Trace hinzu. Abgeleitete Segmente stellen nachgelagerte Dienste und Ressourcen in der Trace-Map dar.

X-Ray bietet ein JSON-Schema für Segmentdokumente. Sie können das Schema hier herunterladen: xray-segmentdocument-schema-v1.0.0. Die im Schema angegebenen Felder und Objekte werden in den folgenden Abschnitten genauer beschrieben.

Eine Teilmenge von Segmentfeldern wird von X-Ray für die Verwendung mit Filterausdrücken indexiert. Wenn Sie beispielsweise das `user` Feld in einem Segment auf eine eindeutige Kennung festlegen, können Sie in der X-Ray-Konsole oder mithilfe der `GetTraceSummaries` API nach Segmenten suchen, die bestimmten Benutzern zugeordnet sind. Weitere Informationen finden Sie unter Verwenden von Filterausdrücken.

Wenn Sie Ihre Anwendung mit dem X-Ray SDK instrumentieren, generiert das SDK Segmentdokumente für Sie. Anstatt Segmentdokumente direkt an X-Ray zu senden, überträgt das SDK sie über einen lokalen UDP-Port an den X-Ray-Daemon. Weitere Informationen finden Sie unter Segmentdokumente an den X-Ray-Daemon senden.

Sections

- Segmentfelder
- Untersegmente

- HTTP-Anfragedaten
- Anmerkungen
- Metadaten
- AWS Ressourcendaten
- Fehler und Ausnahmen
- SQL-Abfragen

Segmentfelder

Ein Segment zeichnet Ablaufverfolgungsinformationen über eine Anforderung auf, die Ihrer Anwendung dient. Ein Segment zeichnet mindestens den Namen, die ID, die Anfangszeit, die Ablaufverfolgungs-ID und die Endzeit der Anforderung auf.

Example Minimales vollständiges Segment

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

Die folgenden Felder sind für Segmente erforderlich oder in einigen Fällen notwendig.

> **Note**
>
> Der Wert muss aus Zeichenfolgen (bis zu 250 Zeichen) bestehen, soweit nicht anders angegeben.

Erforderliche Segmentfelder

- `name`— Der logische Name des Dienstes, der die Anfrage bearbeitet hat, bis zu 200 Zeichen. Beispielsweise der Name der Anwendung oder Domäne. Namen dürfen Unicode-Buchstaben, -Ziffern und -Leerzeichen enthalten sowie die folgenden Symbole: `_`, `.`, `:`, `/`, `%`, `&`, `#`, `=`, `+`, `\`, `-`, `@`
- `id`— Eine 64-Bit-ID für das Segment, die unter den Segmenten in derselben Spur eindeutig ist, mit 16 Hexadezimalziffern.

- `trace_id`— Eine eindeutige Kennung, die alle Segmente und Untersegmente verbindet, die aus einer einzigen Client-Anfrage stammen.

  Röntgen-Trace-ID-Format

  Ein X-Ray `trace_id` besteht aus drei Zahlen, die durch Bindestriche getrennt sind. Beispiel, `1-58406520-a006649127e371903a2de979`. Dies umfasst:

  - Die Versionsnummer, die ist. 1
  - Die Uhrzeit der ursprünglichen Anfrage in Unix-Epochenzeit unter Verwendung von 8 Hexadezimalziffern.

    Zum Beispiel ist 10:00 Uhr am 1. Dezember 2016 PST in Epochenzeit `1480615200` Sekunden oder `58406520` in Hexadezimalziffern.
  - Eine weltweit eindeutige 96-Bit-ID für den Trace mit 24 Hexadezimalziffern.

  > **Note**
  >
  > X-Ray unterstützt jetzt IDs Traces, die mit OpenTelemetry und jedem anderen Framework erstellt wurden, das der W3C Trace Context-Spezifikation entspricht. Eine W3C-Trace-ID muss beim Senden an X-Ray im X-Ray-Trace-ID-Format formatiert werden. Beispielsweise `4efaaf4d1e8720b39541901950019ee5` sollte die W3C-Trace-ID wie `1-4efaaf4d-1e8720b39541901950019ee5` beim Senden an X-Ray formatiert werden. X-Ray-Trace IDs enthält den ursprünglichen Anforderungszeitstempel in der Unix-Epochenzeit, dies ist jedoch nicht erforderlich, wenn W3C-Trace IDs im X-Ray-Format gesendet wird.

  > **Sicherheit der Ablaufverfolgungs-ID**
  >
  > Trace ist in IDs den Headern der Antworten sichtbar. Generieren Sie Trace IDs mit einem sicheren Zufallsalgorhythmus, um sicherzustellen, dass Angreifer future IDs Traces nicht berechnen und IDs damit Anfragen an Ihre Anwendung senden können.

- `start_time`— Zahl, die die Zeit angibt, zu der das Segment erstellt wurde, in Fließkommasekunden in der Epochenzeit. Zum Beispiel `1480615200.010` oder `1.480615200010E9`. Verwenden Sie so viele Dezimalstellen, wie Sie benötigen. Mikrosekundenauflösung ist empfohlen, wenn verfügbar.

- `end_time`— Zahl, die die Zeit angibt, zu der das Segment geschlossen wurde. Zum Beispiel `1480615200.090` oder `1.480615200090E9`. Geben Sie entweder `end_time` oder `in_progress` an.
- `in_progress`— boolescher Wert, auf gesetzt, `true` anstatt anzugeben`end_time`, dass ein Segment gestartet, aber noch nicht abgeschlossen ist. Wenn Ihre Anwendung eine Anforderung empfängt, die lange dauern wird, senden Sie ein in Bearbeitung befindliches Segment, um den Empfang zu bestätigen. Wenn die Antwort gesendet wird, senden Sie das vollständige Segment zum Überschreiben des in Bearbeitung befindlichen Segments. Senden Sie pro Anforderung nur ein vollständiges Segment und ein oder kein angefangenes Segment.

> **Namen der Dienste**
>
> Der eines Segments `name` sollte mit dem Domainnamen oder dem logischen Namen des Dienstes übereinstimmen, der das Segment generiert. Dies wird jedoch nicht erzwungen. Jede Anwendung, die über die entsprechende Berechtigung verfügt, `PutTraceSegments`kann Segmente mit einem beliebigen Namen senden.

Die folgenden Felder sind für Segmente optional.

Optionale Segmentfelder

- `service`— Ein Objekt mit Informationen zu Ihrer Anwendung.
  - `version`— Eine Zeichenfolge, die die Version Ihrer Anwendung identifiziert, die die Anfrage bearbeitet hat.
- `user`— Eine Zeichenfolge, die den Benutzer identifiziert, der die Anfrage gesendet hat.
- `origin`— Der Typ der AWS Ressource, auf der Ihre Anwendung ausgeführt wird.

  Unterstützte Werte

  - `AWS::EC2::Instance`— Eine EC2 Amazon-Instance.
  - `AWS::ECS::Container`— Ein Amazon ECS-Container.
  - `AWS::ElasticBeanstalk::Environment`— Eine Elastic Beanstalk Beanstalk-Umgebung.

  Wenn mehrere Werte für Ihre Anwendung gelten, verwenden Sie den spezifischsten. In einer Multicontainer Docker Elastic Beanstalk Beanstalk-Umgebung wird Ihre Anwendung beispielsweise auf einem Amazon ECS-Container ausgeführt, der wiederum auf einer Amazon-Instance läuft.

EC2 In diesem Fall müssen Sie den Ursprung auf `AWS::ElasticBeanstalk::Environment` festlegen, da die Umgebung das übergeordnete Element der beiden anderen Ressourcen ist.

- `parent_id`— Eine Subsegment-ID, die Sie angeben, wenn die Anfrage von einer instrumentierten Anwendung stammt. Das X-Ray-SDK fügt die ID des übergeordneten Untersegments zum Tracing-Header für Downstream-HTTP-Aufrufe hinzu. Bei verschachtelten Untersegmenten kann ein Untersegment ein Segment oder ein Untersegment als übergeordnetes Element haben.
- `http`— `http`Objekte mit Informationen über die ursprüngliche HTTP-Anfrage.
- `aws`— `aws`Objekt mit Informationen über die AWS Ressource, auf der Ihre Anwendung die Anfrage bearbeitet hat.
- `error`, `throttle`, `fault`, und `cause` — Fehlerfelder, die darauf hinweisen, dass ein Fehler aufgetreten ist, und die Informationen über die Ausnahme enthalten, die den Fehler verursacht hat.
- `annotations`— `annotations`Objekt mit Schlüssel-Wert-Paaren, die X-Ray für die Suche indexieren soll.
- `metadata`— `metadata`Objekt mit allen zusätzlichen Daten, die Sie in dem Segment speichern möchten.
- `subsegments`— Anordnung von `subsegment`Objekten.

Untersegmente

Sie können Untersegmente erstellen, um Aufrufe AWS-Services und Ressourcen, die Sie mit dem AWS SDK tätigen, Aufrufe interner oder externer HTTP-Web APIs - oder SQL-Datenbankabfragen aufzuzeichnen. Sie können auch Untersegmente erstellen, um Code-Blöcke in Ihrer Anwendung zu debuggen oder zu kommentieren. Untersegmente können weitere Untersegmente enthalten, damit ein individuelles Untersegment, das Metadaten über einen internen Funktionsaufruf aufzeichnet, weitere individuelle Untersegmente sowie Untersegmente nachgelagerter Aufrufe umfassen kann.

Ein Untersegment zeichnet einen Downstream-Aufruf aus der Sicht des Dienstes auf, der ihn aufruft. X-Ray verwendet Untersegmente, um Downstream-Services zu identifizieren, die keine Segmente senden, und Einträge für sie im Service-Graph zu erstellen.

Ein Untersegment kann in einem vollständigen Segmentdokument eingebettet oder separat gesendet werden. Senden Sie Untersegmente separat, um nachgelagerte Aufrufe für lange andauernde Anforderungen asynchron nachzuverfolgen oder um zu verhindern, dass die maximale Größe des Segmentdokuments überschritten wird.

Example Segment mit eingebettetem Untersegment

Ein unabhängiges Untersegment hat einen `type` vom `subsegment` und einen `parent_id`, der das übergeordnete Segment identifiziert.

```
{
  "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
        }
      }
    }
  ]
```

```
}
```

Bei Anfragen mit langer Laufzeit können Sie ein Segment in Bearbeitung senden, um X-Ray darüber zu informieren, dass die Anfrage eingegangen ist, und dann Untersegmente separat senden, um sie zu verfolgen, bevor die ursprüngliche Anfrage abgeschlossen wird.

Example In Bearbeitung befindliches Segment

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

Example Unabhängiges Untersegment

Ein unabhängiges Untersegment hat einen `type-subsegment`, eine `trace_id` und eine `parent_id`, die das übergeordnete Segment identifiziert.

```
{
  "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
      }
  }
}
```

Schließen Sie das Segment nach Abschluss der Anforderung mit einer `end_time`. Das vollständige Segment überschreibt das sich in Bearbeitung befindliche Segment.

Sie können außerdem Untersegmente für abgeschlossene Anforderungen, die asynchrone Workflows auslösen, getrennt senden. Beispielsweise kann eine Web-API eine `OK 200`-Antwort, unmittelbar bevor die vom Benutzer angeforderte Arbeit beginnt, zurücksenden. Sie können ein vollständiges Segment an X-Ray senden, sobald die Antwort gesendet wurde, gefolgt von Untersegmenten für später abgeschlossene Arbeiten. Wie bei Segmenten können Sie auch einen Untersegmentsfragment senden, um den Beginn des Untersegments aufzuzeichnen, und es anschließend mit einem vollständigen Segment überschreiben, sobald ein nachgelagerter Aufruf abgeschlossen ist.

Die folgenden Felder sind für Untersegmente erforderlich oder in einigen Fällen notwendig.

> **Note**
>
> Werte sind Zeichenfolgen, die aus bis zu 250 Zeichen bestehen, soweit nicht anders angegeben.

Erforderliche Untersegmentsfelder

- `id`— Eine 64-Bit-ID für das Untersegment, die unter den Segmenten in derselben Spur eindeutig ist und aus 16 Hexadezimalziffern besteht.
- `name`— Der logische Name des Untersegments. Benennen Sie bei nachgelagerten Aufrufen das Untersegment, nachdem die Ressource oder der Service aufgerufen wurde. Benennen Sie bei benutzerdefinierten Untersegmenten das Untersegment nach dem genutzten Code (z. B. einem Funktionsnamen).
- `start_time`— Zahl, die die Zeit angibt, zu der das Untersegment erstellt wurde, in Gleitkommasekunden in der Epochenzeit, auf Millisekunden genau. Zum Beispiel `1480615200.010` oder `1.480615200010E9`.
- `end_time`— Zahl, die die Zeit angibt, zu der das Untersegment geschlossen wurde. Zum Beispiel `1480615200.090` oder `1.480615200090E9`. Geben Sie eine `end_time` oder `in_progress` ein.
- `in_progress`— boolescher Wert, der auf gesetzt ist, `true` anstatt anzugeben, dass ein `end_time` Untersegment zwar gestartet, aber noch nicht abgeschlossen ist. Senden Sie pro

nachgelagerter Anfrage nur ein vollständiges Untersegment und ein oder kein angefangenes Untersegment.

- `trace_id`— Trace-ID des übergeordneten Segments des Untersegments. Nur erforderlich, wenn ein Untersegment separat gesendet wird.

  Röntgen-Trace-ID-Format

  Ein X-Ray `trace_id` besteht aus drei Zahlen, die durch Bindestriche getrennt sind. Beispiel, `1-58406520-a006649127e371903a2de979`. Dies umfasst:

  - Die Versionsnummer, die ist. 1
  - Die Uhrzeit der ursprünglichen Anfrage in Unix-Epochenzeit unter Verwendung von 8 Hexadezimalziffern.

    Zum Beispiel ist 10:00 Uhr am 1. Dezember 2016 PST in Epochenzeit `1480615200` Sekunden oder `58406520` in Hexadezimalziffern.
  - Eine weltweit eindeutige 96-Bit-ID für den Trace mit 24 Hexadezimalziffern.

  > **Note**
  >
  > X-Ray unterstützt jetzt IDs Traces, die mit OpenTelemetry und jedem anderen Framework erstellt wurden, das der W3C Trace Context-Spezifikation entspricht. Eine W3C-Trace-ID muss beim Senden an X-Ray im X-Ray-Trace-ID-Format formatiert werden. Beispielsweise `4efaaf4d1e8720b39541901950019ee5` sollte die W3C-Trace-ID wie `1-4efaaf4d-1e8720b39541901950019ee5` beim Senden an X-Ray formatiert werden. X-Ray-Trace IDs enthält den ursprünglichen Anforderungszeitstempel in der Unix-Epochenzeit, dies ist jedoch nicht erforderlich, wenn W3C-Trace IDs im X-Ray-Format gesendet wird.

- `parent_id`— Segment-ID des übergeordneten Segments des Untersegments. Nur erforderlich, wenn ein Untersegment separat gesendet wird. Bei verschachtelten Untersegmenten kann ein Untersegment ein Segment oder ein Untersegment als übergeordnetes Element haben.
- `type`—`subsegment`. Nur erforderlich, wenn ein Untersegment separat gesendet wird.

Die folgenden Felder sind für Untersegmente optional.

Optionale Untersegmentsfelder

- `namespace`— `aws` für AWS-SDK-Aufrufe; `remote` für andere Downstream-Aufrufe.
- `http`— `http`Objekt mit Informationen über einen ausgehenden HTTP-Aufruf.
- `aws`— `aws`Objekt mit Informationen über die AWS Downstream-Ressource, die Ihre Anwendung aufgerufen hat.
- `error`, `throttle`fault, und `cause` — Fehlerfelder, die darauf hinweisen, dass ein Fehler aufgetreten ist, und die Informationen über die Ausnahme enthalten, die den Fehler verursacht hat.
- `annotations`— `annotations`Objekt mit Schlüssel-Wert-Paaren, die X-Ray für die Suche indexieren soll.
- `metadata`— `metadata`Objekt mit allen zusätzlichen Daten, die Sie in dem Segment speichern möchten.
- `subsegments`— Anordnung von `subsegment`Objekten.
- `precursor_ids`— ein Array von Untersegmenten IDs , das Untersegmente identifiziert, denen dasselbe übergeordnete Objekt zugewiesen wurde, das vor diesem Untersegment abgeschlossen wurde.

HTTP-Anfragedaten

Verwenden Sie einen HTTP-Block, um Details zu einer HTTP-Anforderung aufzuzeichnen, die Ihrer Anwendung (in einem Segment) dient oder die Ihre Anwendung (in einem Untersegment) an eine nachgelagerte HTTP-API gestellt hat. Die meisten Felder in dieser Objektübersicht gehören zu Informationen von HTTP-Anforderungen und -Antworten.

`http`

Alle Felder sind optional.

- `request`— Informationen zu einer Anfrage.
  - `method`— Die Anforderungsmethode. Beispiel, `GET`.
  - `url`— Die vollständige URL der Anfrage, zusammengestellt aus dem Protokoll, dem Hostnamen und dem Pfad der Anfrage.
  - `user_agent`— Die User-Agent-Zeichenfolge vom Client des Anforderers.
  - `client_ip`— Die IP-Adresse des Anforderers. Kann im IP-Paket `Source Address` oder, für weitergeleitete Anforderungen, in einem `X-Forwarded-For`-Header eingesehen werden.

  - `x_forwarded_for`— (nur Segmente) boolescher Wert, der angibt, dass der aus einem `X-Forwarded-For` Header gelesen `client_ip` wurde und nicht zuverlässig ist, da er gefälscht worden sein könnte.
  - `traced`— (nur Untersegmente) boolescher Wert, der angibt, dass der Downstream-Aufruf an einen anderen verfolgten Dienst gerichtet ist. Wenn dieses Feld auf gesetzt ist`true`, betrachtet X-Ray den Trace als unterbrochen, bis der Downstream-Service ein Segment hochlädt`parent_id`, dessen a dem `id` des Untersegments entspricht, das diesen Block enthält.
- `response`— Informationen über eine Antwort.
  - `status`— Ganzzahl, die den HTTP-Status der Antwort angibt.
  - `content_length`— Ganzzahl, die die Länge des Antworttextes in Byte angibt.

Wenn Sie einen Aufruf einer Downstream-Web-API instrumentieren, zeichnen Sie ein Untersegment mit Informationen zur HTTP-Anfrage und -Antwort auf. X-Ray verwendet das Untersegment, um ein abgeleitetes Segment für die Remote-API zu generieren.

Example Segment für HTTP-Aufrufe, die von einer auf Amazon laufenden Anwendung bedient werden 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 Untersegment für einen nachgelagerten HTTP-Aufruf

```
{
  "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 Abgeleitetes Segment für einen nachgelagerten HTTP-Anruf

```
{
  "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
}
```

Anmerkungen

Segmente und Untersegmente können ein `annotations` Objekt enthalten, das ein oder mehrere Felder enthält, die X-Ray für die Verwendung mit Filterausdrücken indexiert. Felder können eine Zeichenfolge, Zahl oder einen booleschen Werte (keine Objekte oder Arrays) umfassen. X-Ray indexiert bis zu 50 Anmerkungen pro Spur.

Example Segment für HTTP-Aufrufe mit Anmerkungen

```
{
  "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
    }
  }
```

Die Schlüssel müssen alphanumerisch sein, um mit Filtern funktionieren zu können. Unterstriche sind zulässig. Andere Zeichen und Leerzeichen sind nicht zulässig.

Metadaten

Segmente und Untersegmente können ein `metadata` Objekt enthalten, das ein oder mehrere Felder mit Werten beliebigen Typs, einschließlich Objekten und Arrays, enthält. X-Ray indexiert keine Metadaten, und Werte können beliebig groß sein, solange das Segmentdokument die maximale Größe (64 kB) nicht überschreitet. Sie können Metadaten im vollständigen Segmentdokument, das von der `BatchGetTraces`-API zurückgesendet wurde, einsehen. Feldschlüssel (`debug`im folgenden Beispiel), die mit beginnen, `AWS.` sind für die Verwendung durch AWS-provided SDKs und Clients reserviert.

Example Individuelles Untersegment mit Metadaten

```
{
  "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"
        ]
      }
    }
  ]
}
```

AWS Ressourcendaten

Bei Segmenten enthält das `aws`-Objekt Informationen zu den Ressourcen, auf denen Ihre Anwendung ausgeführt wird. Mehrere Felder können für eine einzelne Ressource zutreffen. Beispielsweise könnte eine Anwendung, die in einer Docker-Umgebung mit mehreren Containern auf Elastic Beanstalk ausgeführt wird, Informationen über die EC2 Amazon-Instance, den Amazon ECS-Container, der auf der Instance ausgeführt wird, und die Elastic Beanstalk Beanstalk-Umgebung selbst enthalten.

`aws` (Segmente)

Alle Felder sind optional.

- `account_id`— Wenn Ihre Anwendung Segmente an eine andere sendet AWS-Konto, notieren Sie sich die ID des Kontos, auf dem Ihre Anwendung ausgeführt wird.
- `cloudwatch_logs`— Array von Objekten, die eine einzelne CloudWatch Protokollgruppe beschreiben.
  - `log_group`— Der Name der CloudWatch Protokollgruppe.
  - `arn`— Die CloudWatch Protokollgruppe ARN.
- `ec2`— Informationen über eine EC2 Amazon-Instance.
  - `instance_id`— Die Instance-ID der EC2 Instance.
  - `instance_size`— Der Typ der EC2 Instanz.

  - `ami_id`— Die Amazon Machine Image-ID.
  - `availability_zone`— Die Availability Zone, in der die Instance ausgeführt wird.
- `ecs`— Informationen über einen Amazon ECS-Container.
  - `container`— Der Hostname Ihres Containers.
  - `container_id`— Die vollständige Container-ID Ihres Containers.
  - `container_arn`— Der ARN Ihrer Container-Instance.
- `eks`— Informationen über einen Amazon EKS-Cluster.
  - `pod`— Der Hostname Ihres EKS-Pods.
  - `cluster_name`— Der Name des EKS-Clusters.
  - `container_id`— Die vollständige Container-ID Ihres Containers.
- `elastic_beanstalk`— Informationen über eine Elastic Beanstalk Beanstalk-Umgebung. Sie finden diese Informationen in einer Datei mit dem Namen `/var/elasticbeanstalk/xray/environment.conf` auf den neuesten Elastic Beanstalk-Plattformen.
  - `environment_name` – Der Name der Umgebung.
  - `version_label`— Der Name der Anwendungsversion, die derzeit auf der Instance bereitgestellt wird, die die Anfrage bedient hat.
  - `deployment_id`— Zahl, die die ID der letzten erfolgreichen Bereitstellung auf der Instance angibt, die die Anfrage bedient hat.
- `xray`— Metadaten über den Typ und die Version der verwendeten Instrumentierung.
  - `auto_instrumentation`— Boolescher Wert, der angibt, ob die automatische Instrumentierung verwendet wurde (z. B. der Java-Agent).
  - `sdk_version`— Die Version des verwendeten SDK oder Agenten.
  - `sdk`— Der Typ des SDK.

Example AWS Block mit Plugins

```
"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"
   }
}
```

Erfassen Sie für Untersegmente Informationen über die Ressourcen AWS-Services und die Ressourcen, auf die Ihre Anwendung zugreift. X-Ray verwendet diese Informationen, um abgeleitete Segmente zu erstellen, die die Downstream-Services in Ihrer Service-Map darstellen.

`aws` (Untersegmente)

Alle Felder sind optional.

- `operation`— Der Name der API-Aktion, die für eine AWS-Service OR-Ressource aufgerufen wurde.
- `account_id`— Wenn Ihre Anwendung auf Ressourcen in einem anderen Konto zugreift oder Segmente an ein anderes Konto sendet, notieren Sie sich die ID des Kontos, dem die AWS Ressource gehört, auf die Ihre Anwendung zugegriffen hat.
- `region`— Wenn sich die Ressource in einer anderen Region als Ihre Anwendung befindet, notieren Sie die Region. Beispiel, `us-west-2`.
- `request_id`— Eindeutiger Bezeichner für die Anfrage.
- `queue_url`— Für Operationen in einer Amazon SQS SQS-Warteschlange die URL der Warteschlange.
- `table_name`— Für Operationen an einer DynamoDB-Tabelle der Name der Tabelle.

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements

```
{
```

```
  "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",
  }
}
```

Fehler und Ausnahmen

Wenn ein Fehler auftritt, können Sie die Einzelheiten zum Fehler und den Ausnahmen, die er generiert, aufzeichnen. Zeichnen Sie Fehler in Segmenten auf, wenn Ihre Anwendung einen Fehler an den Benutzer zurückgibt, sowie in Untersegmenten, wenn ein nachgelagerter Aufruf einen Fehler ausgibt.

Fehlertypen

Stellen Sie ein oder mehrere der folgenden Felder auf `true` ein, um anzuzeigen, dass ein Fehler aufgetreten ist. Bei schwerwiegenden Fehlern können mehrere Typen ausgewählt werden. Ein `429 Too Many Requests`-Fehler von einem nachgelagerten Aufruf kann beispielsweise dazu führen, dass Ihre Anwendung zu einem `500 Internal Server Error` zurückkehrt. In diesem Fall treffen alle drei Typen zu.

- `error`— Boolescher Wert, der angibt, dass ein Client-Fehler aufgetreten ist (der Antwortstatuscode lautete 4XX Client Error).
- `throttle`— boolescher Wert, der angibt, dass eine Anfrage gedrosselt wurde (der Antwortstatuscode lautete 429 Too Many Requests).
- `fault`— boolescher Wert, der angibt, dass ein Serverfehler aufgetreten ist (der Antwortstatuscode lautete 5XX Server Error).

Geben Sie die Fehlerursache an, indem Sie im Segment oder Untersegment ein Ursachenobjekt einschließen.

`cause`

Eine Ursache kann entweder eine 16-stellige Ausnahmen-ID oder ein Objekt mit den folgenden Feldern sein:

- `working_directory`— Der vollständige Pfad des Arbeitsverzeichnisses, als die Ausnahme auftrat.
- `paths`— Das Array von Pfaden zu Bibliotheken oder Modulen, die verwendet wurden, als die Ausnahme auftrat.
- `exceptions`— Das Array von Ausnahmeobjekten.

Geben Sie detaillierte Informationen über die Fehler in einem oder mehreren Ausnahmenobjekten an.

`exception`

Alle Felder sind optional.

- `id`— Eine 64-Bit-ID für die Ausnahme, die unter den Segmenten derselben Spur eindeutig ist und aus 16 Hexadezimalziffern besteht.
- `message`— Die Ausnahmemeldung.
- `type`— Der Ausnahmetyp.
- `remote`— Boolescher Wert, der angibt, dass die Ausnahme durch einen Fehler verursacht wurde, der von einem nachgelagerten Dienst zurückgegeben wurde.
- `truncated`— Ganzzahl, die die Anzahl der Stack-Frames angibt, die in der weggelassen wurden. `stack`
- `skipped`— Ganzzahl, die die Anzahl der Ausnahmen angibt, die zwischen dieser Ausnahme und ihrem untergeordneten Element, d. h. der von ihr verursachten Ausnahme, übersprungen wurden.
- `cause`— Ausnahme-ID des der Ausnahme übergeordneten Elements, d. h. der Ausnahme, die diese Ausnahme verursacht hat.
- `stack`— Array von StackFrame-Objekten.

Falls verfügbar, zeichnen Sie Informationen zu dem Aufruf-Stack in stackFrame-Objekten auf.

`stackFrame`

Alle Felder sind optional.

- `path`— Der relative Pfad zur Datei.
- `line`— Die Zeile in der Datei.
- `label`— Der Funktions- oder Methodenname.

SQL-Abfragen

Für Anfragen, die Ihre Anwendung in eine SQL-Datenbank umwandeln, können Sie Untersegmente erstellen.

`sql`

Alle Felder sind optional.

- `connection_string`— Für SQL Server- oder andere Datenbankverbindungen, die keine URL-Verbindungszeichenfolgen verwenden, notieren Sie sich die Verbindungszeichenfolge ohne Kennwörter.
- `url`— Notieren Sie für eine Datenbankverbindung, die eine URL-Verbindungszeichenfolge verwendet, die URL ohne Kennwörter.
- `sanitized_query`— Die Datenbankabfrage, bei der alle vom Benutzer angegebenen Werte entfernt oder durch einen Platzhalter ersetzt wurden.
- `database_type`— Der Name der Datenbank-Engine.
- `database_version`— Die Versionsnummer der Datenbank-Engine.
- `driver_version`— Der Name und die Versionsnummer des Datenbank-Engine-Treibers, den Ihre Anwendung verwendet.
- `user`— Der Datenbank-Benutzername.
- `preparation`— `call` ob die Abfrage eine verwendet hat`PreparedCall`; `statement` wenn die Abfrage eine verwendet hat`PreparedStatement`.

Example Untersegment mit einer SQL-Abfrage

```
{
  "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-Ray Konzepte

AWS X-Ray empfängt Daten von Diensten als Segmente. X-Ray gruppiert dann Segmente, die eine gemeinsame Anforderung haben, in Traces. X-Ray verarbeitet die Traces, um ein Service-Diagramm zu erstellen, das eine visuelle Darstellung Ihrer Anwendung bietet.

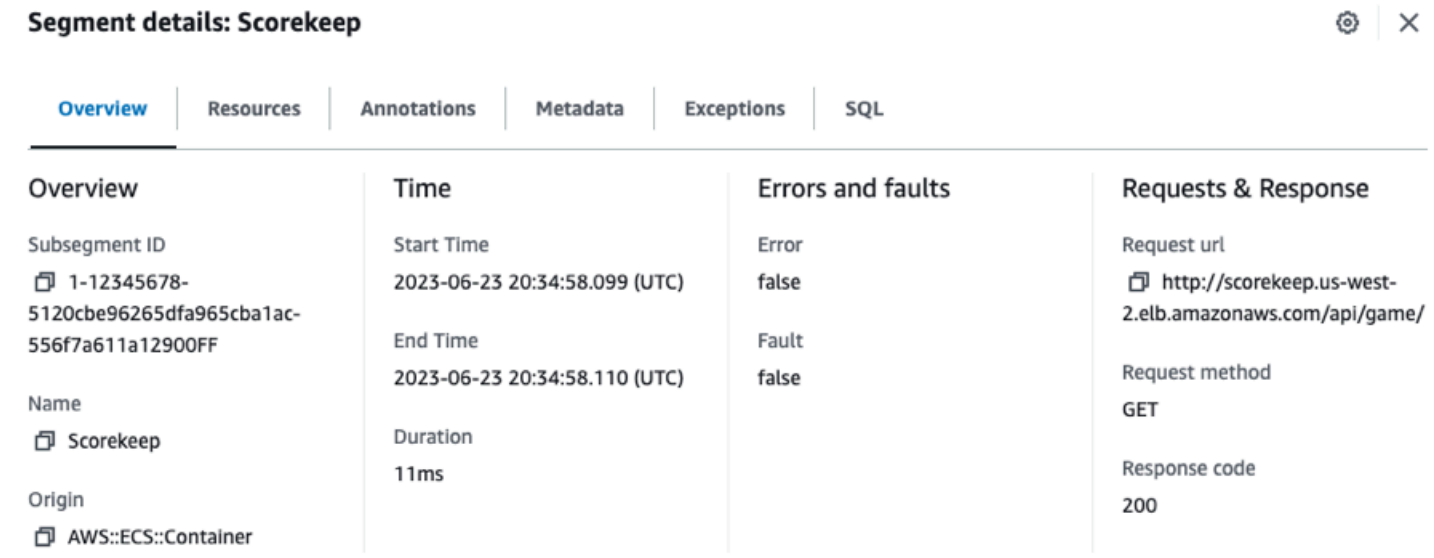
Konzepte

- Segmente
- Untersegmente
- Service-Diagramm
- Ablaufverfolgungen
- Sampling
- Ablaufverfolgungs-Header
- Filterausdrücke
- Gruppen
- Anmerkungen und Metadaten
- Fehler und Ausnahmen

Segmente

Die Datenverarbeitungsressourcen, die Ihre Anwendungslogik ausführen, senden Daten zu ihrer Arbeit als Segment. Ein Segment enthält den Namen der Ressource, Details zu der Anforderung und Details zu den erledigten Aufgaben. Wenn zum Beispiel eine HTTP-Anforderung Ihre Anwendung erreicht, können Daten über Folgendes erfasst werden:

- Der Host — Hostname, Alias oder IP-Adresse
- Die Anfrage — Methode, Client-Adresse, Pfad, Benutzeragent
- Die Antwort — Status, Inhalt
- Die geleistete Arbeit — Start- und Endzeiten, Untersegmente
- Auftretende Probleme — Fehler, Störungen und Ausnahmen, einschließlich der automatischen Erfassung von Ausnahmestapeln.



Das X-Ray SDK sammelt Informationen aus Anfrage- und Antwort-Headern, dem Code in Ihrer Anwendung und Metadaten über die AWS Ressourcen, auf denen es ausgeführt wird. Sie wählen die zu erfassenden Daten aus, indem Sie Ihre Anwendungskonfiguration oder Ihren Code ändern, um eingehende Anfragen, Downstream-Anfragen und AWS SDK-Clients zu instrumentieren.

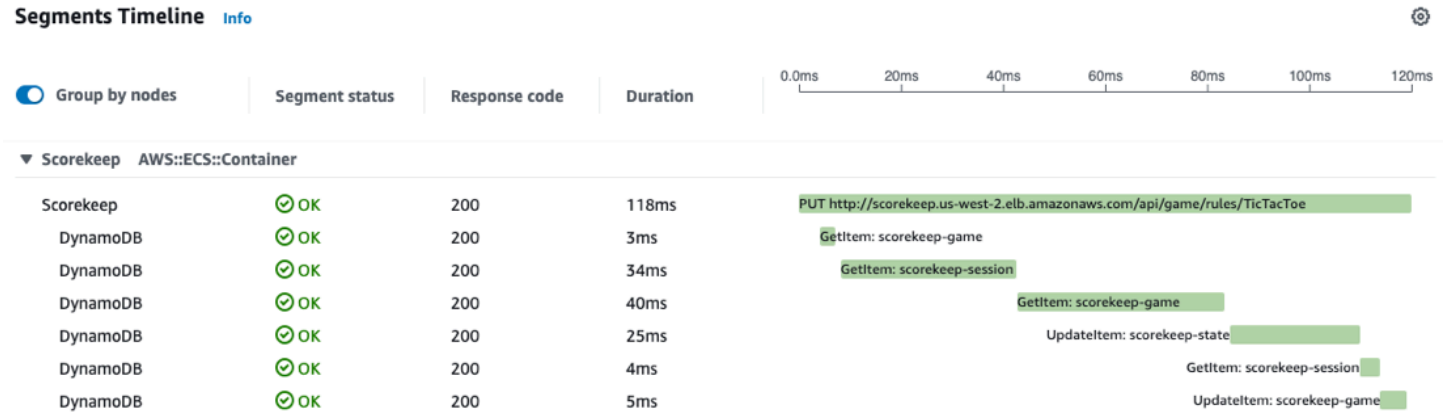
> **Weitergeleitete Anfragen**
>
> Wenn ein Load Balancer oder ein anderer Vermittler eine Anfrage an Ihre Anwendung weiterleitet, nimmt X-Ray die Client-IP aus dem `X-Forwarded-For` Header in der Anfrage und nicht aus der Quell-IP im IP-Paket. Die Client-IP, die für eine weitergeleitete Anfrage aufgezeichnet wird, kann gefälscht sein und sollte daher nicht als vertrauenswürdig eingestuft werden.

Sie können das X-Ray SDK verwenden, um zusätzliche Informationen wie Anmerkungen und Metadaten aufzuzeichnen. Weitere Details über die Struktur und die Informationen, die in Segmenten und Untersegmenten aufgezeichnet werden, finden Sie unter AWS X-Ray Dokumente segmentieren. Segmentdokumente können bis zu 64 kB groß sein.

Untersegmente

Ein Segment kann die Daten zu der geleisteten Arbeit in Untersegmente unterteilen. Untersegmente bieten detailliertere Zeitinformationen und Einzelheiten zu Downstream-Aufrufen, die Ihre Anwendung getätigt hat, um die ursprüngliche Anforderung zu bearbeiten. Ein Untersegment kann zusätzliche

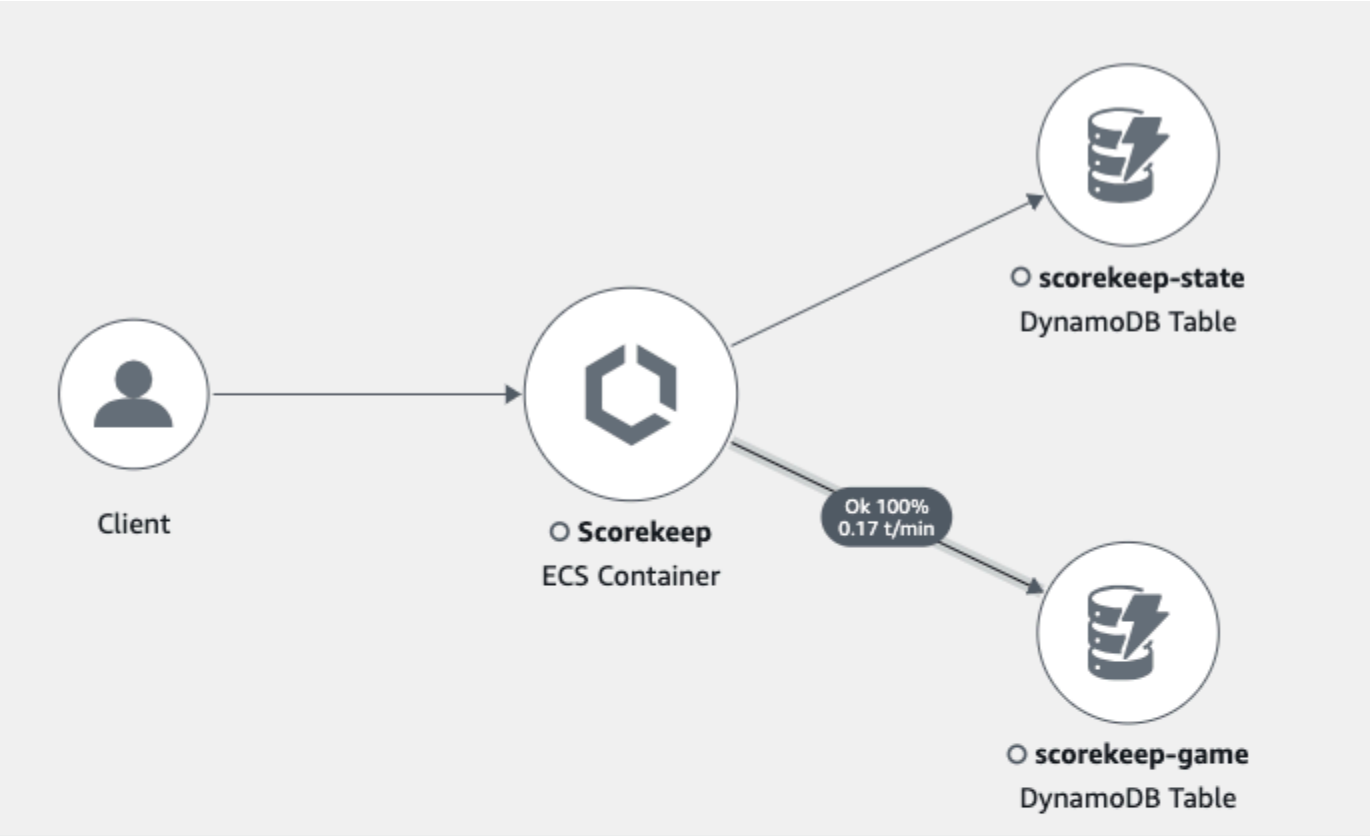
Details zu einem Aufruf einer AWS-Service, einer externen HTTP-API oder einer SQL-Datenbank enthalten. Sie können sogar beliebige Untersegmente definieren, um spezifische Funktionen oder Codezeilen in Ihrer Anwendung zu instrumentieren.

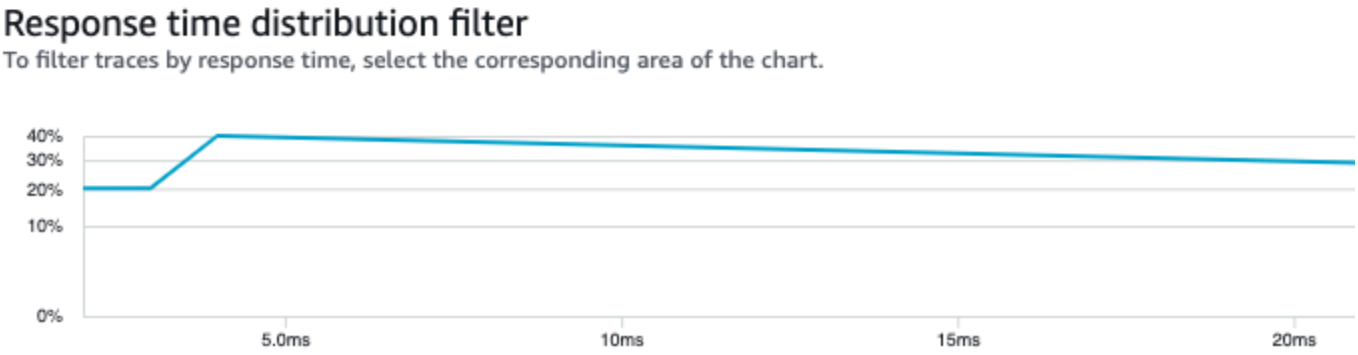


Für Dienste, die keine eigenen Segmente senden, wie Amazon DynamoDB, verwendet X-Ray Untersegmente, um abgeleitete Segmente und Downstream-Knoten auf der Trace-Map zu generieren. Dadurch können Sie alle Ihre nachgelagerten Abhängigkeiten einsehen, auch wenn diese keine Ablaufverfolgung unterstützen oder externe Ressourcen sind.

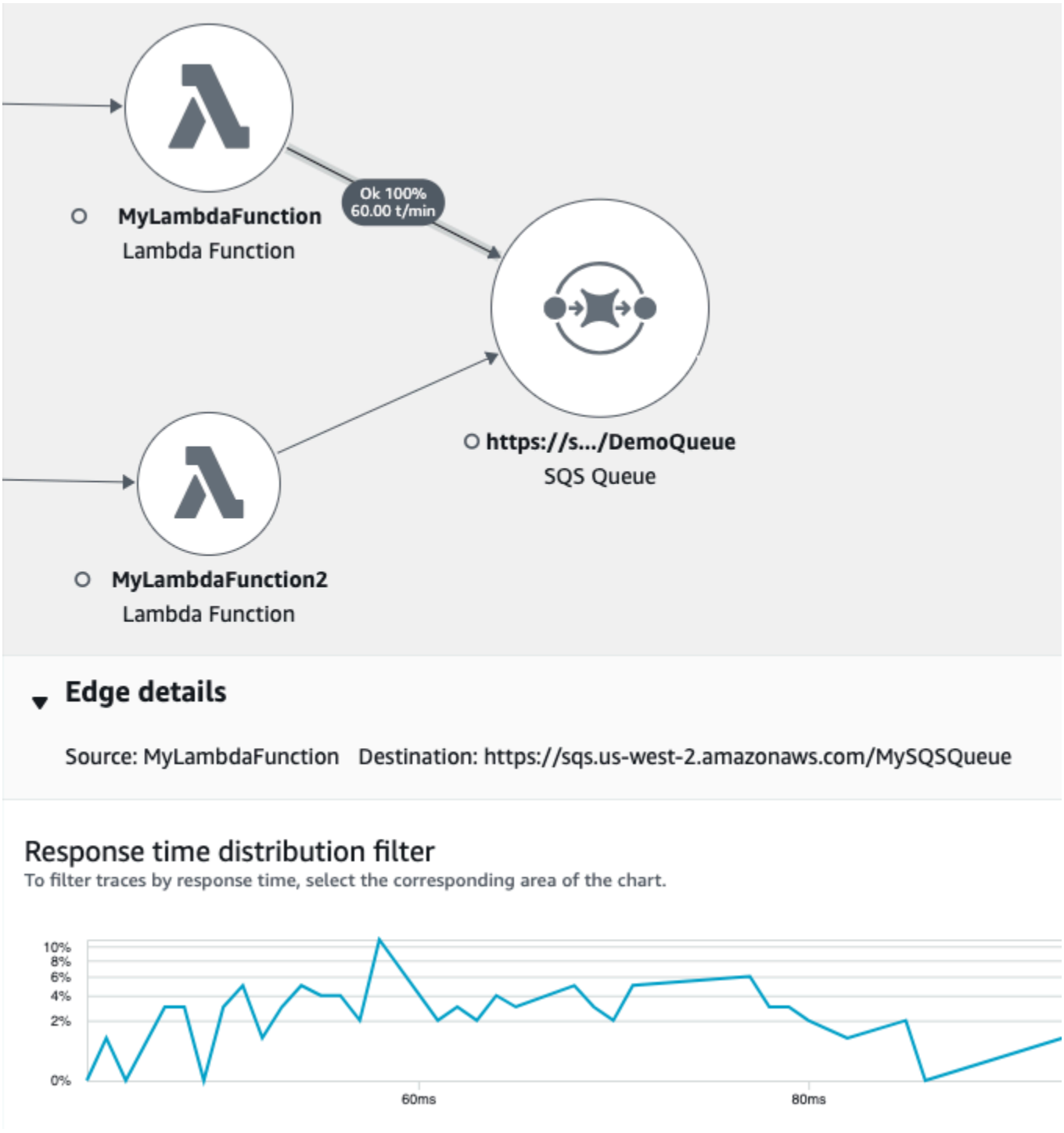
Untersegmente geben die Ansicht eines nachgelagerten Aufrufs Ihrer Anwendung als Client wieder. Wenn der nachgelagerte Service ebenfalls instrumentiert ist, ersetzt das von ihm gesendete Segment das abgeleitete Segment, das aus dem Untersegment des vorgelagerten Clients generiert wurde. Der Knoten im Service-Diagramm verwendet immer Informationen vom Segment des Services, sofern verfügbar, während die Edge zwischen den beiden Knoten das Untersegment des nachgelagerten Services verwendet.

Wenn Sie DynamoDB beispielsweise mit einem instrumentierten AWS SDK-Client aufrufen, zeichnet das X-Ray-SDK ein Untersegment für diesen Aufruf auf. DynamoDB sendet kein Segment, sodass das abgeleitete Segment im Trace, der DynamoDB-Knoten im Service-Graph und die Kante zwischen Ihrem Service und DynamoDB alle Informationen aus dem Subsegment enthalten.





Wenn Sie einen weiteren instrumentierten Service mit einer instrumentierten Anwendung aufrufen, sendet der nachgelagerte Service ein eigenes Segment, um seine Ansicht desselben Aufrufs aufzuzeichnen, den der vorgelagerte Service in einem Untersegment aufgezeichnet hat. Im Service-Diagramm enthalten die Knoten beider Services Zeit- und Fehlerinformationen aus den Segmenten dieser Services, während die Edge zwischen ihnen Informationen aus dem Untersegment des vorgelagerten Services enthält.



Beide Ansichten sind nützlich, da der nachgelagerte Service genau aufzeichnet, wann die Verarbeitung der Anfrage gestartet und beendet wurde, und der vorgelagerten Service zeichnet die Umlauflatenz auf, einschließlich der Zeit für die Übertragung der Latenz zwischen den beiden Services.

Service-Diagramm

X-Ray verwendet die Daten, die Ihre Anwendung sendet, um ein Service-Diagramm zu generieren. Jede AWS Ressource, die Daten an X-Ray sendet, wird im Diagramm als Dienst angezeigt. Grenzen verbinden die Services, die gemeinsam Anforderungen bearbeiten. Edges verbinden Clients mit Ihrer Anwendung und Ihre Anwendung mit den Downstream-Services sowie den verwendeten Ressourcen.

> **Namen der Dienste**
>
> Der eines Segments `name` sollte mit dem Domainnamen oder dem logischen Namen des Dienstes übereinstimmen, der das Segment generiert. Dies wird jedoch nicht erzwungen. Jede Anwendung, die dazu berechtigt ist, `PutTraceSegments`kann Segmente mit einem beliebigen Namen senden.

Eine Service-Graphik ist ein JSON-Dokument, das Informationen zu den Services und Ressourcen enthält, aus denen Ihre Anwendung besteht. Die X-Ray-Konsole verwendet den Service Graph, um eine Visualisierung oder Service Map zu generieren.

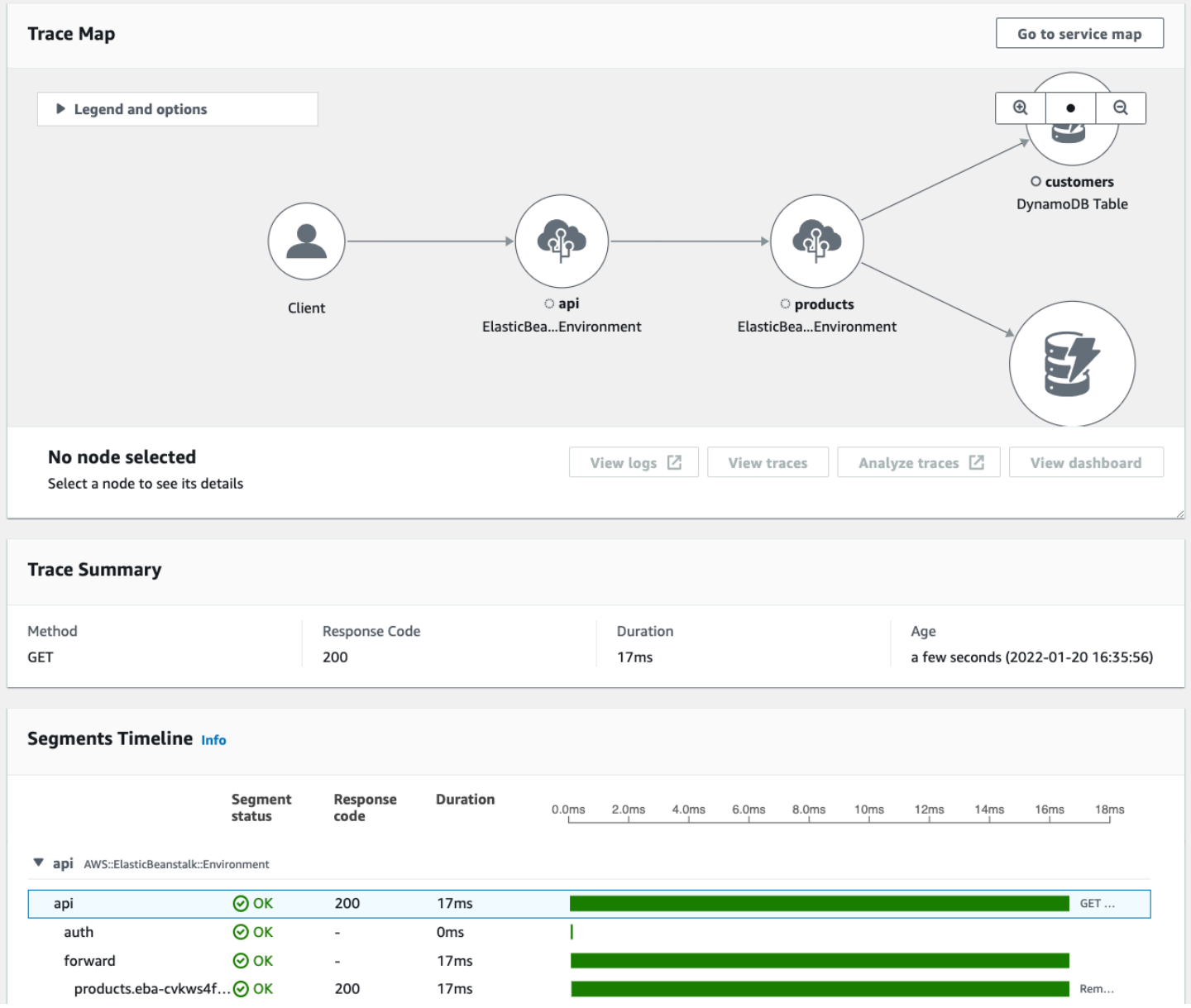
Für eine verteilte Anwendung kombiniert X-Ray Knoten aller Dienste, die Anfragen mit derselben Trace-ID verarbeiten, in einem einzigen Service-Graphen. Der erste Service, den die Anforderung findet, fügt einen Nachverfolgungskopf hinzu, der zwischen dem Frontend und den davon aufgerufenen Services verteilt wird.

Beispielsweise führt Scorekeep eine Web-API aus, die einen Mikroservice (eine AWS Lambda -Funktion) aufruft, die mithilfe einer Node.js-Bibliothek einen zufälligen Namen generiert. Das X-Ray SDK for Java generiert die Trace-ID und nimmt sie in Lambda-Aufrufe auf. Lambda sendet Ablaufverfolgungsdaten und übergibt die Trace-ID an die Funktion. Das X-Ray SDK für Node.js verwendet auch die Trace-ID, um Daten zu senden. Daher werden Knoten für die API, den Lambda-Service und die Lambda-Funktion alle als separate, aber verbundene Knoten auf der Trace-Map angezeigt.

Die Daten des Service-Diagramms werden 30 Tage lang aufbewahrt.

Ablaufverfolgungen

Eine Ablaufverfolgungs-ID verfolgt den Weg einer Anforderung durch Ihre Anwendung. Eine Ablaufverfolgung sammelt alle von einer einzelnen Anforderung generierten Segmente. Bei dieser Anfrage handelt es sich in der Regel um eine HTTP-GET- oder POST-Anforderung, die einen Load Balancer durchläuft, auf Ihren Anwendungscode trifft und Downstream-Aufrufe an andere AWS Dienste oder externe Websites APIs generiert. Der erste unterstützte Service, mit dem die HTTP-Anforderungen interagiert, fügt der Anforderung einen Ablaufverfolgungs-ID-Kopf hinzu und verteilt ihn nachgeordnet weiter, um Latenz, Disposition und andere Daten der Anforderung nachzuverfolgen.



Informationen zur AWS X-Ray Abrechnung von Röntgen-Traces finden Sie unter Preise. Trace-Daten werden 30 Tage lang aufbewahrt.

Sampling

Um eine effiziente Ablaufverfolgung zu gewährleisten und eine repräsentative Stichprobe der Anfragen bereitzustellen, die Ihre Anwendung bearbeitet, wendet das X-Ray SDK einen Sampling-Algorithmus an, um zu bestimmen, welche Anfragen nachverfolgt werden. Standardmäßig zeichnet das X-Ray-SDK jede Sekunde die erste Anfrage und fünf Prozent aller weiteren Anfragen auf.

Um zu vermeiden, dass bei den ersten Schritten Servicegebühren anfallen, ist die Standard-Samplingrate konservativ. Sie können X-Ray so konfigurieren, dass die standardmäßige Stichprobenregel geändert und zusätzliche Regeln konfiguriert werden, die das Sampling auf der Grundlage der Eigenschaften des Dienstes oder der Anfrage anwenden.

Möglicherweise möchten Sie das Sampling deaktivieren und alle Anfragen für Anrufe verfolgen, die den Status ändern oder Benutzer oder Transaktionen verarbeiten. Für große, schreibgeschützte Aufrufe wie Abfragen im Hintergrund, Zustandsprüfungen oder Anschlusswartung können Sie eine niedrige Samplingrate einstellen und erhalten dennoch genügend Daten, um auftretende Probleme zu erkennen.

Weitere Informationen finden Sie unter Konfigurieren von -Samplingregeln.

Ablaufverfolgungs-Header

Alle Anforderungen werden bis zu einem konfigurierbaren Minimum verfolgt. Wenn dieses Minimum erreicht ist, wird ein Prozentsatz der Anforderungen nachverfolgt, um unnötige Kosten zu vermeiden. Die Samplingentscheidung und die Ablaufverfolgungs-ID werden HTTP-Anforderungen in Ablaufverfolgungs-Headern mit der Bezeichnung `X-Amzn-Trace-Id` hinzugefügt. Der erste X-Ray-integrated Dienst, auf den die Anfrage trifft, fügt einen Tracing-Header hinzu, der vom X-Ray SDK gelesen und in die Antwort aufgenommen wird.

Example Ablaufverfolgungs-Header mit Stammablaufverfolgungs-ID und Samplingentscheidung

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

> **Sicherheit des Ablaufverfolgungs-Headers**
>
> Ein Tracing-Header kann vom X-Ray SDK AWS-Service, einer oder der Client-Anfrage stammen. Ihre Anwendung kann `X-Amzn-Trace-Id` aus eingehenden Anfragen entfernen, um Probleme zu vermeiden, die dadurch entstehen, dass Benutzer ihren Anfragen Trace IDs - oder Sampling-Entscheidungen hinzufügen.

Der Ablaufverfolgungs-Header kann auch eine übergeordnete Segment-ID enthalten, wenn die Anforderung von einer instrumentierten Anwendung stammte. Wenn Ihre Anwendung beispielsweise eine Downstream-HTTP-Web-API mit einem instrumentierten HTTP-Client aufruft, fügt das X-Ray-SDK die Segment-ID für die ursprüngliche Anfrage zum Tracing-Header der Downstream-Anfrage hinzu. Eine instrumentierte Anwendung, die die nachgelagerte Anforderung verarbeitet, kann die übergeordnete Segment-ID erfassen, um die beiden Anforderungen zu verbinden.

Example Ablaufverfolgungs-Header mit Stammablaufverfolgungs-ID, übergeordnete Segment-ID und Samplingentscheidung

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

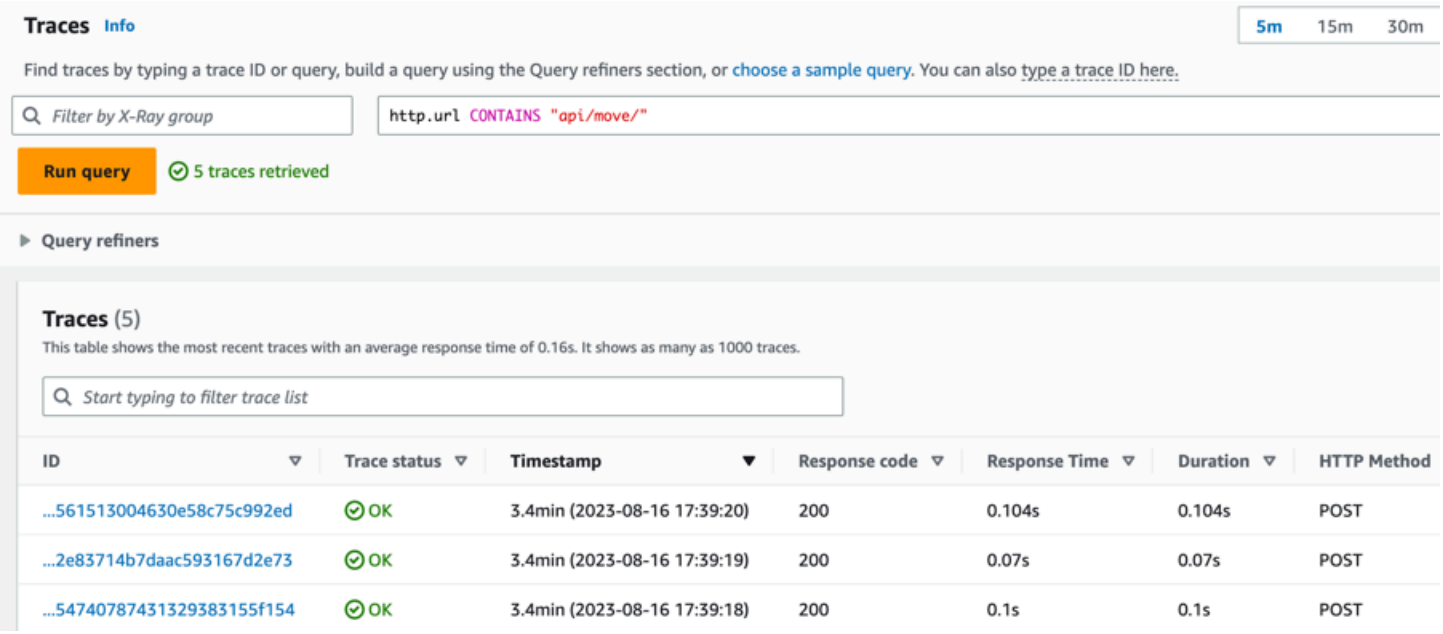
`Lineage`können von Lambda und anderen AWS-Services als Teil ihrer Verarbeitungsmechanismen an den Trace-Header angehängt werden und sollten nicht direkt verwendet werden.

Example Tracing-Header mit Lineage

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

Filterausdrücke

Selbst mit Sampling generiert eine komplexe Anwendung eine Fülle von Daten. Die AWS X-Ray Konsole bietet eine easy-to-navigate Ansicht des Service-Graphen. Sie zeigt Status- und Leistungsinformationen, mit denen Sie Probleme und Möglichkeiten zur Optimierung Ihrer Anwendung ermitteln können. Für die erweiterte Ablaufverfolgung können Sie Details der Ablaufverfolgungen für einzelne Anforderungen anzeigen oder Filterausdrücke verwenden, um Ablaufverfolgungen in Bezug auf bestimmte Pfade oder Benutzer zu suchen.



Gruppen

X-Ray erweitert die Filterausdrücke und unterstützt auch die Gruppenfunktion. Mit einem Filterausdruck können Sie Kriterien definieren, nach denen Ablaufverfolgungen in der Gruppe akzeptiert werden sollen.

Sie können die Gruppe mit ihrem Namen oder mit dem Amazon-Ressourcennamen (ARN) aufrufen, um ihr eigenes Service-Diagramm, Trace-Zusammenfassungen und CloudWatch Amazon-Metriken zu generieren. Sobald eine Gruppe erstellt wurde, werden eingehende Traces mit dem Filterausdruck der Gruppe verglichen, während sie im X-Ray-Dienst gespeichert werden. Metriken für die Anzahl der Traces, die den einzelnen Kriterien entsprechen, werden CloudWatch minutengenau veröffentlicht.

Durch das Aktualisieren des Filterausdrucks einer Gruppe werden die bereits aufgezeichneten Daten nicht geändert. Die Aktualisierung gilt nur für nachfolgende Ablaufverfolgungen. Dies kann dazu führen, dass im Diagramm neue und alte Ausdrücke zusammengeführt werden. Um dies zu vermeiden, löschen Sie die aktuelle Gruppe und erstellen Sie eine neue.

> **Note**
>
> Gruppen werden anhand der Anzahl der abgerufenen Ablaufverfolgungen, die dem Filterausdruck entsprechen, in Rechnung gestellt. Weitere Informationen finden Sie unter AWS X-Ray Preise.

Weitere Informationen zu Gruppen finden Sie unterGruppen konfigurieren.

Anmerkungen und Metadaten

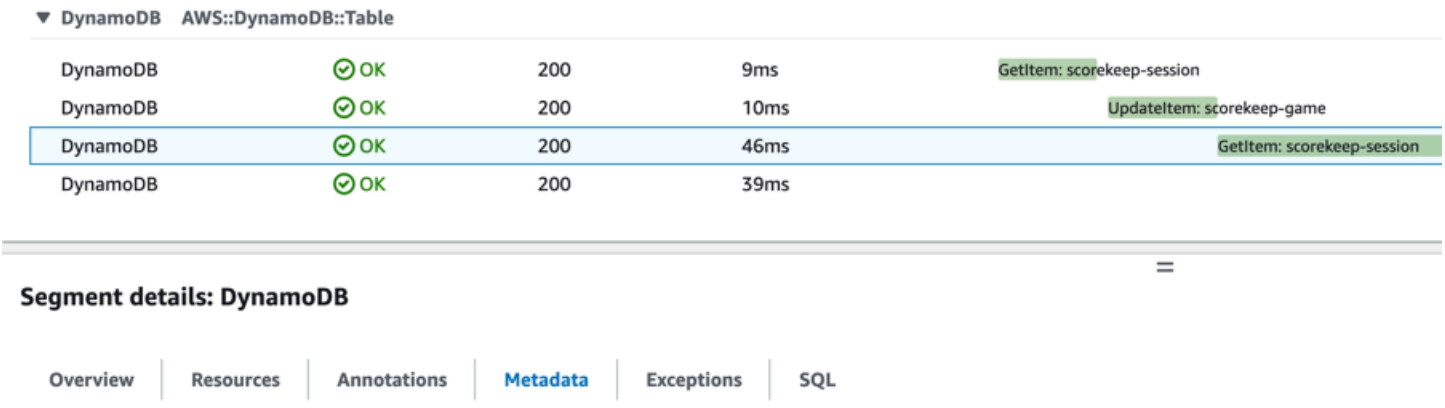
Wenn Sie Ihre Anwendung instrumentieren, zeichnet das X-Ray SDK Informationen über eingehende und ausgehende Anfragen, die verwendeten AWS Ressourcen und die Anwendung selbst auf. Sie können dem Segmentdokument weitere Informationen hinzufügen, wie etwa Anmerkungen und Metadaten. Anmerkungen und Metadaten werden auf der Trace-Ebene aggregiert und können jedem Segment oder Teilsegment hinzugefügt werden.

Anmerkungen sind einfache Schlüssel-Wert-Paare, die zur Verwendung mit Filterausdrücken indiziert sind. Berücksichtigen Sie Anmerkungen, um Daten zur Gruppierung von Ablaufverfolgungen in der Konsole zu verwenden, oder wenn Sie die `GetTraceSummaries`-API aufrufen.

X-Ray indexiert bis zu 50 Anmerkungen pro Spur.

Metadaten sind Schlüssel-Wert-Paare mit Werten aller Art (darunter Objekte und Listen), jedoch ohne Indizierung. Verwenden Sie Metadaten zum Aufzeichnen von Daten in der Ablaufverfolgung, die nicht zur Ablaufsuche erforderlich sind.

Sie können Anmerkungen und Metadaten im Fenster mit den Segment- oder Untersegmentdetails auf der Seite mit den Trace-Details in der Konsole anzeigen. CloudWatch



Fehler und Ausnahmen

X-Ray verfolgt Fehler, die in Ihrem Anwendungscode auftreten, und Fehler, die von nachgelagerten Diensten zurückgegeben werden. Fehler sind wie folgt kategorisiert.

- **`Error`**— Client-Fehler (Fehler der Serie 400)

- **`Fault`**— Serverfehler (Fehler der Serie 500)
- **`Throttle`**— Drosselungsfehler (429 zu viele Anfragen)

Wenn eine Ausnahme auftritt, während Ihre Anwendung eine instrumentierte Anfrage bearbeitet, zeichnet das X-Ray SDK Details zu der Ausnahme auf, einschließlich des Stack-Trace, falls verfügbar. Sie können Ausnahmen unter Segmentdetails in der X-Ray-Konsole anzeigen.

Sicherheit in AWS X-Ray

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS-Services in der läuft AWS Cloud. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS -Compliance-Programms getestet und überprüft. Weitere Informationen zu den Compliance-Programmen, die für X-Ray gelten, finden Sie unter AWS-Services Umfang nach Compliance-Programmen.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS-Service , was Sie verwenden. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von X-Ray anwenden können. In den folgenden Themen erfahren Sie, wie Sie X-Ray konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere verwenden können AWS-Services , die Ihnen helfen können, Ihre X-Ray-Ressourcen zu überwachen und zu sichern.

Topics

Datenschutz in AWS X-Ray

AWS X-Ray verschlüsselt immer Traces und zugehörige Daten im Ruhezustand. Wenn Sie Verschlüsselungsschlüssel aus Compliance-Gründen oder internen Anforderungen prüfen und deaktivieren müssen, können Sie X-Ray so konfigurieren, dass ein AWS Key Management Service (AWS KMS) -Schlüssel zum Verschlüsseln von Daten verwendet wird.

X-Ray bietet einen Von AWS verwalteter Schlüssel Name`aws/xray`. Verwenden Sie diesen Schlüssel, wenn Sie die Schlüsselverwendung in AWS CloudTrail prüfen möchten, den eigentlichen Schlüssel aber nicht verwalten müssen. Wenn Sie den Zugriff auf den Schlüssel verwalten oder die Schlüsselrotation konfigurieren müssen, können Sie einen vom Kunden verwalteten Schlüssel erstellen.

Wenn Sie die Verschlüsselungseinstellungen ändern, verbringt X-Ray einige Zeit damit, Datenschlüssel zu generieren und zu verbreiten. Während der neue Schlüssel verarbeitet wird, kann X-Ray Daten mit einer Kombination aus den neuen und alten Einstellungen verschlüsseln. Vorhandene Daten werden nicht erneut verschlüsselt, wenn Sie Verschlüsselungseinstellungen ändern.

> **Note**
>
> AWS KMS berechnet Gebühren, wenn X-Ray einen KMS-Schlüssel zum Verschlüsseln oder Entschlüsseln von Trace-Daten verwendet.
>
> - Standardverschlüsselung — Kostenlos.
> - Von AWS verwalteter Schlüssel— Zahlen Sie für die Nutzung des Schlüssels.
> - vom Kunden verwalteter Schlüssel — Zahlen Sie für die Aufbewahrung und Nutzung der Schlüssel.
>
> Einzelheiten finden Sie unter AWS Key Management Service Preise.

> **Note**
>
> X-Ray Insights-Benachrichtigungen senden Ereignisse an Amazon EventBridge, das derzeit keine vom Kunden verwalteten Schlüssel unterstützt. Weitere Informationen finden Sie unter Datenschutz bei Amazon EventBridge.

Sie benötigen Zugriff auf Benutzerebene auf einen vom Kunden verwalteten Schlüssel, um X-Ray für dessen Verwendung zu konfigurieren und dann verschlüsselte Traces anzeigen zu können. Weitere Informationen finden Sie unter Benutzerberechtigungen für die Verschlüsselung.

CloudWatch console

So konfigurieren Sie X-Ray für die Verwendung eines KMS-Schlüssels für die Verschlüsselung mithilfe der CloudWatch Konsole

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich Einstellungen aus.
3. Wählen Sie im Bereich X-Ray-Traces unter Verschlüsselung die Option Einstellungen anzeigen.
4. Wählen Sie im Bereich Verschlüsselungskonfiguration die Option Bearbeiten aus.
5. Wählen Sie KMS-Schlüssel verwenden aus.
6. Wählen Sie einen Schlüssel aus dem Dropdown-Menü aus:
   - aws/xray — Verwenden Sie die. Von AWS verwalteter Schlüssel
   - Schlüsselalias — Verwenden Sie einen vom Kunden verwalteten Schlüssel in Ihrem Konto.
   - Manuelles Eingeben eines Schlüssel-ARN — Verwenden Sie einen vom Kunden verwalteten Schlüssel in einem anderen Konto. Geben Sie den vollständigen Amazon-Ressourcennamen (ARN) des Schlüssels in das entsprechende Feld ein.
7. Wählen Sie Verschlüsselung aktualisieren.

X-Ray console

So konfigurieren Sie X-Ray für die Verwendung eines KMS-Schlüssels für die Verschlüsselung mithilfe der X-Ray-Konsole

1. Öffnen Sie die X-Ray-Konsole.
2. Wählen Sie Encryption (Verschlüsselung) aus.
3. Wählen Sie KMS-Schlüssel verwenden.
4. Wählen Sie einen Schlüssel aus dem Dropdown-Menü aus:
   - aws/xray — Verwenden Sie die. Von AWS verwalteter Schlüssel

- Schlüsselalias — Verwenden Sie einen vom Kunden verwalteten Schlüssel in Ihrem Konto.
- Manuelles Eingeben eines Schlüssel-ARN — Verwenden Sie einen vom Kunden verwalteten Schlüssel in einem anderen Konto. Geben Sie den vollständigen Amazon-Ressourcennamen (ARN) des Schlüssels in das entsprechende Feld ein.

5. Wählen Sie Anwenden aus.

> **Note**
>
> X-Ray unterstützt keine asymmetrischen KMS-Schlüssel.

Wenn X-Ray nicht auf Ihren Verschlüsselungsschlüssel zugreifen kann, werden keine Daten mehr gespeichert. Dies kann passieren, wenn Ihr Benutzer den Zugriff auf den KMS-Schlüssel verliert oder wenn Sie einen Schlüssel deaktivieren, der gerade verwendet wird. In diesem Fall zeigt X-Ray eine Benachrichtigung in der Navigationsleiste an.

Informationen zur Konfiguration von Verschlüsselungseinstellungen mit der X-Ray-API finden Sie unterKonfigurieren von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray -API.

Identitäts- und Zugriffsmanagement für AWS X-Ray

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um X-Ray-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Funktionen zugreifen können (sieheFehlerbehebung bei AWS X-Ray Identität und Zugriff)
- Dienstadministrator – bestimmen Sie den Benutzerzugriff und reichen Sie Berechtigungsanfragen ein (sieheWie AWS X-Ray funktioniert mit IAM)
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (sieheAWS X-Ray Beispiele für identitätsbasierte Richtlinien)

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Verwenden Sie möglichst temporäre Anmeldeinformationen anstelle von IAM-

Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer für den Zugriff AWS mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für Verbundbenutzerzugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, dienstübergreifenden Zugriff und Anwendungen, die auf Amazon ausgeführt werden. EC2 Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an Identitäten oder Ressourcen anhängen. AWS Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die daraus resultierenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie AWS X-Ray funktioniert mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf X-Ray zu verwalten, sollten Sie wissen, welche IAM-Funktionen für die Verwendung mit X-Ray verfügbar sind. Einen allgemeinen Überblick darüber, wie X-Ray und andere Produkte mit IAM AWS-Services funktionieren AWS-Services , finden Sie unter That Work with IAM im IAM-Benutzerhandbuch.

Sie können AWS Identity and Access Management (IAM) verwenden, um Benutzern X-Ray-Berechtigungen zu gewähren und Ressourcen in Ihrem Konto zu berechnen. IAM steuert den Zugriff auf den X-Ray-Dienst auf API-Ebene, um Berechtigungen einheitlich durchzusetzen, unabhängig davon, welchen Client (Konsole, AWS SDK AWS CLI) Ihre Benutzer verwenden.

Um die X-Ray-Konsole zum Anzeigen von Trace-Maps und Segmenten zu verwenden, benötigen Sie lediglich Leseberechtigungen. Um den Zugriff auf die Konsole zu ermöglichen, fügen Sie Ihrem IAM-Benutzer die `AWSXrayReadOnlyAccess` verwaltete Richtlinie hinzu.

Erstellen Sie für lokale Entwicklungen und Tests eine IAM-Rolle mit Lese- und Schreibberechtigungen. Übernehmen Sie die Rolle und speichern Sie temporäre

Anmeldeinformationen für die Rolle. Sie können diese Anmeldeinformationen mit dem X-Ray-Daemon AWS CLI, dem und dem AWS SDK verwenden. Weitere Informationen finden Sie unter Verwenden temporärer Sicherheitsanmeldedaten mit dem AWS CLI.

Um Ihre instrumentierte App bereitzustellen AWS, erstellen Sie eine IAM-Rolle mit Schreibberechtigungen und weisen Sie sie den Ressourcen zu, auf denen Ihre Anwendung ausgeführt wird. `AWSXRayDaemonWriteAccess`beinhaltet die Erlaubnis, Traces hochzuladen, und einige Leseberechtigungen, um die Verwendung von Sampling-Regeln zu unterstützen.

Die Lese- und Schreibrichtlinien enthalten keine Berechtigung zum Konfigurieren von Einstellungen für den Verschlüsselungsschlüssel und Samplingregeln. Wird verwendet, `AWSXrayFullAccess` um auf diese Einstellungen zuzugreifen oder eine Konfiguration APIs in einer benutzerdefinierten Richtlinie hinzuzufügen. Für Verschlüsselung und Entschlüsselung mit einem von Ihnen erstellten kundenverwalteten Schlüssel benötigen Sie auch die Berechtigung zur Verwendung des Schlüssels.

Themen

- Identitätsbasierte X-Ray-Richtlinien
- Ressourcenbasierte X-Ray-Richtlinien
- Autorisierung auf Basis von X-Ray-Tags
- Lokale Ausführung Ihrer Anwendung
- Ihre Anwendung wird ausgeführt in AWS
- Benutzerberechtigungen für die Verschlüsselung

Identitätsbasierte X-Ray-Richtlinien

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. X-Ray unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Richtlinienaktionen in X-Ray verwenden das folgende Präfix vor der Aktion:`xray:`. Um beispielsweise jemandem die Erlaubnis zu erteilen, Gruppenressourcendetails mit dem `GetGroup` X-Ray-API-Vorgang abzurufen, nehmen Sie die `xray:GetGroup` Aktion in seine Richtlinie auf. Richtlinienanweisungen müssen entweder ein – `Action`oder ein `NotAction`-Element enthalten. X-Ray definiert eigene Aktionen, die Aufgaben beschreiben, die Sie mit diesem Dienst ausführen können.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

```
"Action": [
      "xray:action1",
      "xray:action2"
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Get` beginnen, einschließlich der folgenden Aktion:

```
"Action": "xray:Get*"
```

Eine Liste der X-Ray-Aktionen finden Sie unter Definierte Aktionen von AWS X-Ray im IAM-Benutzerhandbuch.

Ressourcen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie bei Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Sie können den Zugriff auf Ressourcen mithilfe einer IAM-Richtlinie steuern. Für Aktionen, die Berechtigungen auf Ressourcenebene unterstützen, verwenden Sie einen Amazon-Ressourcennamen (ARN), um die Ressource zu identifizieren, für die die Richtlinie gilt.

Alle X-Ray-Aktionen können in einer IAM-Richtlinie verwendet werden, um Benutzern die Erlaubnis zur Verwendung dieser Aktion zu erteilen oder zu verweigern. Allerdings unterstützen nicht alle X-Ray-Aktionen Berechtigungen auf Ressourcenebene, mit denen Sie angeben können, für welche Ressourcen eine Aktion ausgeführt werden kann.

Für Aktionen, die Berechtigungen auf Ressourcenebene nicht unterstützen, muss „*“ als Ressource verwendet werden.

Die folgenden X-Ray-Aktionen unterstützen Berechtigungen auf Ressourcenebene:

- `CreateGroup`
- `GetGroup`
- `UpdateGroup`
- `DeleteGroup`
- `CreateSamplingRule`
- `UpdateSamplingRule`
- `DeleteSamplingRule`

Nachstehend finden Sie ein Beispiel für eine identitätsbasierte Berechtigungsrichtlinie für eine `CreateGroup`-Aktion: Das Beispiel zeigt die Verwendung eines ARN in Bezug auf den Gruppennamen `local-users` mit der eindeutigen ID als Platzhalter. Die eindeutige ID wird generiert, wenn die Gruppe erstellt wird, und kann daher in der Richtlinie nicht im Voraus vorhergesehen werden. Bei der Verwendung von `GetGroup`, `UpdateGroup` oder `DeleteGroup` können Sie diese entweder als Platzhalter oder als den genauen ARN definieren, einschließlich ID.

> **Note**
>
> Der ARN einer Sampling-Regel definiert sich anhand ihres Namens. Im Gegensatz zu Gruppen ARNs haben Stichprobenregeln keine eindeutig generierte ID.

Eine Liste der X-Ray-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie AWS X-Ray im IAM-Benutzerhandbuch unter Defined by (Ressourcen definiert von). Informationen zu den Aktionen,

mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von AWS X-Ray definierte Aktionen.

Bedingungsschlüssel

X-Ray stellt keine dienstspezifischen Bedingungsschlüssel bereit, unterstützt aber die Verwendung einiger globaler Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Beispiele

Beispiele für identitätsbasierte X-Ray-Richtlinien finden Sie unter. AWS X-Ray Beispiele für identitätsbasierte Richtlinien

Ressourcenbasierte X-Ray-Richtlinien

X-Ray unterstützt ressourcenbasierte Richtlinien für die aktuelle und future AWS-Service Integration, wie Amazon SNS Active Tracing. Ressourcenbasierte X-Ray-Richtlinien können durch andere AWS-Managementkonsole s oder über das AWS SDK oder die CLI aktualisiert werden. Die Amazon SNS SNS-Konsole versucht beispielsweise, automatisch ressourcenbasierte Richtlinien für das Senden von Traces an X-Ray zu konfigurieren. Das folgende Richtliniendokument enthält ein Beispiel für die manuelle Konfiguration ressourcenbasierter X-Ray-Richtlinien.

Example Beispiel für eine ressourcenbasierte X-Ray-Richtlinie für Amazon SNS Active Tracing

Dieses Beispielrichtliniendokument spezifiziert die Berechtigungen, die Amazon SNS benötigt, um Trace-Daten an X-Ray zu senden:

```
{
    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"
          }
        }
      }
    ]
  }
```

Verwenden Sie die CLI, um eine ressourcenbasierte Richtlinie zu erstellen, die Amazon SNS Berechtigungen zum Senden von Trace-Daten an X-Ray erteilt:

```
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" } } } ] }'
```

Um diese Beispiele zu verwenden, ersetzen Sie*partition*, *regionaccount-id*, und *topic-name* durch Ihre spezifische AWS Partition, Region, Konto-ID und Ihren Amazon SNS SNS-Themennamen. Um allen Amazon SNS SNS-Themen die Erlaubnis zu geben, Trace-Daten an X-Ray zu senden, ersetzen Sie den Themennamen durch. *

Autorisierung auf Basis von X-Ray-Tags

Sie können Tags an X-Ray-Gruppen oder Sampling-Regeln anhängen oder Tags in einer Anfrage an X-Ray übergeben. Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `xray:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden. Weitere Informationen zum Taggen von X-Ray-Ressourcen finden Sie unterKennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Verwaltung des Zugriffs auf X-Ray-Gruppen und Stichprobenregeln auf der Grundlage von Tags.

Lokale Ausführung Ihrer Anwendung

Ihre instrumentierte Anwendung sendet Trace-Daten an den X-Ray-Daemon. Der Daemon puffert segmentierte Dokumente und lädt sie stapelweise in den X-Ray-Dienst hoch. Der Daemon benötigt Schreibberechtigungen, um Trace-Daten und Telemetrie in den X-Ray-Dienst hochzuladen.

Wenn Sie den Daemon lokal ausführen, erstellen Sie eine IAM-Rolle, übernehmen Sie die Rolle und speichern Sie temporäre Anmeldeinformationen in Umgebungsvariablen oder in einer Datei, die `credentials` in einem Ordner mit dem Namen Ihres Benutzerordners benannt `.aws` ist. Weitere Informationen finden Sie unter Verwenden temporärer Sicherheitsanmeldedaten mit dem 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}
```

Wenn Sie bereits Anmeldeinformationen für die Verwendung mit dem AWS SDK oder konfiguriert haben AWS CLI, kann der Daemon diese verwenden. Wenn mehrere Profile verfügbar sind, verwendet der Daemon das Standard-Profil.

Ihre Anwendung wird ausgeführt in AWS

Wenn Sie Ihre Anwendung auf ausführen AWS, verwenden Sie eine Rolle, um der EC2 Amazon-Instance oder Lambda-Funktion, die den Daemon ausführt, Berechtigungen zu erteilen.

- Amazon Elastic Compute Cloud (Amazon EC2) — Erstellen Sie eine IAM-Rolle und fügen Sie sie der EC2 Instance als Instance-Profil hinzu.
- Amazon Elastic Container Service (Amazon ECS) — Erstellen Sie eine IAM-Rolle und hängen Sie sie als Container-Instance-IAM-Rolle an Container-Instances an.
- AWS Elastic Beanstalk (Elastic Beanstalk) — Elastic Beanstalk enthält X-Ray-Berechtigungen in seinem Standard-Instanzprofil. Sie können das Standard-Instance-Profil verwenden oder einem benutzerdefinierten Instance-Profil Schreibberechtigungen hinzufügen.
- AWS Lambda (Lambda) — Fügen Sie der Ausführungsrolle Ihrer Funktion Schreibberechtigungen hinzu.

Um eine Rolle für die Verwendung mit X-Ray zu erstellen

1. Öffnen Sie die IAM-Konsole.
2. Wählen Sie Roles.
3. Klicken Sie auf Create New Role.
4. Geben Sie für Role Name (Name der Rolle) **`xray-application`** ein. Wählen Sie Next Step (Weiter) aus.
5. Wählen Sie als Rollentyp Amazon aus EC2.
6. Fügen Sie die folgende verwaltete Richtlinie hinzu, auf die Ihre Anwendung zugreifen kann AWS-Services:

   - AWSXRayDaemonWriteAccess— Erlaubt dem X-Ray-Daemon die Erlaubnis, Trace-Daten hochzuladen.

   Wenn Ihre Anwendung das AWS SDK für den Zugriff auf andere Dienste verwendet, fügen Sie Richtlinien hinzu, die den Zugriff auf diese Dienste gewähren.
7. Wählen Sie Next Step (Weiter) aus.
8. Wählen Sie Rolle erstellen aus.

Benutzerberechtigungen für die Verschlüsselung

X-Ray verschlüsselt standardmäßig alle Trace-Daten, und Sie können es so konfigurieren, dass ein von Ihnen verwalteter Schlüssel verwendet wird. Wenn Sie sich für einen vom AWS Key Management Service Kunden verwalteten Schlüssel entscheiden, müssen Sie sicherstellen, dass die Zugriffsrichtlinie des Schlüssels es Ihnen ermöglicht, X-Ray die Erlaubnis zu erteilen, ihn zum Verschlüsseln zu verwenden. Andere Benutzer in Ihrem Konto benötigen ebenfalls Zugriff auf den Schlüssel, um verschlüsselte Trace-Daten in der X-Ray-Konsole anzeigen zu können.

Für einen vom Kunden verwalteten Schlüssel konfigurieren Sie Ihren Schlüssel mit einer Zugriffsrichtlinie, die die folgenden Aktionen ermöglicht:

- Der Benutzer, der den Schlüssel in X-Ray konfiguriert, hat die Berechtigung, `kms:CreateGrant` und `kms:DescribeKey` anzurufen.
- Benutzer, die auf verschlüsselte Ablaufverfolgungsdaten zugreifen können, besitzen die Berechtigung zum Aufruf von `kms:Decrypt`.

Wenn Sie der Gruppe Hauptbenutzer im Bereich Schlüsselkonfiguration der IAM-Konsole einen Benutzer hinzufügen, hat er die erforderlichen Rechte für diese beiden Operationen. Die Berechtigungen müssen nur für die Schlüsselrichtlinie festgelegt werden, sodass Sie keine AWS KMS Berechtigungen für Ihre Benutzer, Gruppen oder Rollen benötigen. Weitere Informationen finden Sie unter Verwenden wichtiger Richtlinien im AWS KMS Entwicklerhandbuch.

Bei der Standardverschlüsselung oder wenn Sie das AWS verwaltete CMK (`aws/xray`) wählen, hängt die Berechtigung davon ab, wer Zugriff auf X-Ray APIs hat. Jeder Benutzer mit Zugriff auf `PutEncryptionConfig`, enthalten in `AWSXrayFullAccess`, kann die Verschlüsselungskonfiguration ändern. Um zu verhindern, dass ein Benutzer den Verschlüsselungsschlüssel ändert, gewähren Sie ihm nicht die Berechtigung zur Verwendung von `PutEncryptionConfig`.

AWS X-Ray Beispiele für identitätsbasierte Richtlinien

Standardmäßig sind Benutzer und Rollen nicht berechtigt, X-Ray-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mit der AWS-Managementkonsole AWS CLI, oder AWS API ausführen. Ein Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator muss diese Richtlinien anschließend den -Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von Richtlinien auf der JSON-Registerkarte im IAM-Benutzerhandbuch.

Themen

- Best Practices für Richtlinien
- Verwenden der X-Ray-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Verwaltung des Zugriffs auf X-Ray-Gruppen und Stichprobenregeln auf der Grundlage von Tags
- Von IAM verwaltete Richtlinien für X-Ray
- X-Ray-Updates für AWS verwaltete Richtlinien
- Angabe einer Ressource innerhalb einer IAM-Richtlinie

Best Practices für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand X-Ray-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Beachten Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.

- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der X-Ray-Konsole

Um auf die AWS X-Ray Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den X-Ray-Ressourcen in Ihrem aufzulisten und anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Um sicherzustellen, dass diese Entitäten weiterhin die X-Ray-Konsole verwenden können, hängen Sie die `AWSXRayReadOnlyAccess` AWS verwaltete Richtlinie an die Entitäten an. Diese Richtlinie wird unter IAM-verwaltete Richtlinien für X-Ray ausführlicher beschrieben. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die den API-Operation entsprechen, die Sie ausführen möchten.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder 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": "*"
        }
    ]
}
```

Verwaltung des Zugriffs auf X-Ray-Gruppen und Stichprobenregeln auf der Grundlage von Tags

Sie können Bedingungen in Ihrer identitätsbasierten Richtlinie verwenden, um den Zugriff auf X-Ray-Gruppen und Stichprobenregeln auf der Grundlage von Stichwörtern zu kontrollieren. Die folgende Beispielrichtlinie könnte verwendet werden, um einer Benutzerrolle die Berechtigungen zum Erstellen, Löschen oder Aktualisieren von Gruppen mit den Tags `stage:prod` oder zu verweigern. `stage:preprod` Weitere Informationen zum Markieren von Regeln und Gruppen für die Röntgenabtastung finden Sie unterKennzeichnen von Regeln und Gruppen für die Röntgenprobenahme.

Um die Erstellung einer Stichprobenregel `aws:RequestTag` zu verweigern, geben Sie damit Tags an, die nicht als Teil einer Erstellungsanforderung übergeben werden können. Um die Aktualisierung oder Löschung einer Stichprobenregel `aws:ResourceTag` zu verweigern, verwenden Sie „Ablehnen“ von Aktionen, die auf den Tags dieser Ressourcen basieren.

Sie können diese Richtlinien den Benutzern in Ihrem Konto zuordnen (oder sie zu einer einzigen Richtlinie zusammenfassen und dann die Richtlinie anhängen). Damit der Benutzer Änderungen an einer Gruppen- oder Stichprobenregel vornehmen kann, darf die Gruppen- oder Stichprobenregel nicht mit `stage=prepod` oder gekennzeichnet sein`stage=prod`. Der Tag-Schlüssel `Stage` der Bedingung stimmt sowohl mit `Stage` als auch mit `stage` überein, da die Namen von Bedingungsschlüsseln nicht zwischen Groß- und Kleinschreibung unterscheiden. Weitere Informationen zum Bedingungsblock finden Sie unter IAM JSON Policy Elements: Condition im IAM-Benutzerhandbuch.

Ein Benutzer mit einer Rolle, der die folgende Richtlinie zugewiesen ist, kann das Tag nicht `role:admin` zu Ressourcen hinzufügen und keine Tags aus einer Ressource entfernen, die `role:admin` damit verknüpft ist.

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"
                }
            }
        }
    ]
}
```

Von IAM verwaltete Richtlinien für X-Ray

Um die Erteilung von Berechtigungen zu vereinfachen, unterstützt IAM verwaltete Richtlinien für jeden Dienst. Ein Service kann diese verwalteten Richtlinien mit neuen Berechtigungen aktualisieren, wenn er neue APIs veröffentlicht. AWS X-Ray stellt verwaltete Richtlinien für nur Lese-, Schreib- und Administratoranwendungsfälle bereit.

- `AWSXrayReadOnlyAccess`— Leseberechtigungen für die Verwendung der X-Ray-Konsole oder des AWS SDK zum Abrufen von Trace-Daten, Trace-Maps, Erkenntnissen und der X-Ray-Konfiguration von der X-Ray-API. AWS CLI Beinhaltet Observability Access Manager (OAM) `oam:ListSinks` und `oam:ListAttachedSinks` Berechtigungen, die es der Konsole ermöglichen, im Rahmen der CloudWatch kontenübergreifenden Observability geteilte Traces von Quellkonten einzusehen. Die `GetDistinctTraceGraphs` API-Aktionen `BatchGetTraceSummaryById` und sind nicht dafür vorgesehen, von Ihrem Code aufgerufen zu werden, und sie sind auch nicht im und enthalten. 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`— Schreibberechtigungen für die Verwendung des X-Ray-Daemons oder AWS SDK zum Hochladen von Segmentdokumenten und Telemetrie in die X-Ray-API. AWS CLI Umfasst Leseberechtigungen zum Abrufen von Samplingregeln und zur Berichterstellung zu Samplingergebnissen.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
```

```
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- `AWSXrayCrossAccountSharingConfiguration`— Erteilt Berechtigungen zum Erstellen, Verwalten und Anzeigen von Observability Access Manager-Links für die gemeinsame Nutzung von X-Ray-Ressourcen zwischen Konten. Wird verwendet, um die CloudWatch kontenübergreifende Observability zwischen Quell- und Monitoring-Konten zu ermöglichen.

  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`— Erlaubnis zur Nutzung aller APIs X-Ray-Inhalte, einschließlich Lese- und Schreibberechtigungen sowie der Erlaubnis, Verschlüsselungsschlüsseleinstellungen und Sampling-Regeln zu konfigurieren. Beinhaltet Observability Access Manager (OAM) `oam:ListSinks` und `oam:ListAttachedSinks` Berechtigungen, die es der Konsole ermöglichen, im Rahmen der CloudWatch kontenübergreifenden Observability geteilte Traces von Quellkonten einzusehen.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:*",
                "oam:ListSinks"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

So fügen Sie einem IAM-Benutzer, einer Gruppe oder einer Rolle eine veraltete Richtlinie hinzu:

1. Öffnen Sie die IAM-Konsole.

2. Öffnen Sie die Rolle, die Ihrem Instance-Profil, einem IAM-Benutzer oder einer IAM-Gruppe zugewiesen ist.
3. Fügen Sie unter Berechtigungen die verwaltete Richtlinie an.

X-Ray-Updates für AWS verwaltete Richtlinien

Sehen Sie sich Details zu Aktualisierungen der AWS verwalteten Richtlinien für X-Ray an, seit dieser Dienst begonnen hat, diese Änderungen zu verfolgen. Um automatische Benachrichtigungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der X-Ray-Dokument-Verlaufsseite.

Änderungen	Beschreibung	Date
IAM-verwaltete Richtlinien für X-Ray — Neue `AWSXrayCrossAccountSharingConfiguration` `AWSXrayReadOnlyAccess` und aktualisierte `AWSXrayFullAccess` Richtlinien hinzugefügt.	X-Ray fügte Observability Access Manager (OAM) - Berechtigungen `oam:ListSinks` und `oam:ListAttachedSinks` zu diesen Richtlinien hinzu, sodass die Konsole im Rahmen der CloudWatch kontoüber greifenden Observability von Quellkonten geteilte Traces anzeigen kann.	27. November 2022
IAM-verwaltete Richtlinien für X-Ray — Aktualisierung der `AWSXrayReadOnlyAccess` Richtlinie.	X-Ray hat eine API-Aktion hinzugefügt,`ListResourcePolicies` .	15. November 2022
Verwenden der X-Ray-Konsole — Aktualisierung der `AWSXrayReadOnlyAccess` Richtlinie	X-Ray hat zwei neue API-Aktionen hinzugefügt, `BatchGetTraceSummaryById` und`GetDistinctTraceGraphs` .	11. November 2022

Änderungen	Beschreibung	Date
	Diese Aktionen sind nicht dafür vorgesehen, von Ihrem Code aufgerufen zu werden. Daher sind diese API-Aktio nen nicht im AWS CLI und enthalten AWS SDKs.	

Angabe einer Ressource innerhalb einer IAM-Richtlinie

Sie können den Zugriff auf Ressourcen mithilfe einer IAM-Richtlinie steuern. Für Aktionen, die Berechtigungen auf Ressourcenebene unterstützen, verwenden Sie einen Amazon-Ressourcennamen (ARN), um die Ressource zu identifizieren, für die die Richtlinie gilt.

Alle X-Ray-Aktionen können in einer IAM-Richtlinie verwendet werden, um Benutzern die Erlaubnis zur Verwendung dieser Aktion zu erteilen oder zu verweigern. Allerdings unterstützen nicht alle X-Ray-Aktionen Berechtigungen auf Ressourcenebene, mit denen Sie angeben können, für welche Ressourcen eine Aktion ausgeführt werden kann.

Für Aktionen, die Berechtigungen auf Ressourcenebene nicht unterstützen, muss „*“ als Ressource verwendet werden.

Die folgenden X-Ray-Aktionen unterstützen Berechtigungen auf Ressourcenebene:

- `CreateGroup`
- `GetGroup`
- `UpdateGroup`
- `DeleteGroup`
- `CreateSamplingRule`
- `UpdateSamplingRule`
- `DeleteSamplingRule`

Nachstehend finden Sie ein Beispiel für eine identitätsbasierte Berechtigungsrichtlinie für eine `CreateGroup`-Aktion: Das Beispiel zeigt die Verwendung eines ARN in Bezug auf den Gruppennamen `local-users` mit der eindeutigen ID als Platzhalter. Die eindeutige ID wird

generiert, wenn die Gruppe erstellt wird, und kann daher in der Richtlinie nicht im Voraus vorhergesehen werden. Bei der Verwendung von `GetGroup`, `UpdateGroup` oder `DeleteGroup` können Sie diese entweder als Platzhalter oder als den genauen ARN definieren, einschließlich ID.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateGroup"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
            ]
        }
    ]
}
```

Nachstehend finden Sie ein Beispiel für eine identitätsbasierte Berechtigungsrichtlinie für eine `CreateSamplingRule`-Aktion:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateSamplingRule"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:sampling-rule/base-
scorekeep"
            ]
        }
    ]
```

```
}
```

> **Note**
>
> Der ARN einer Sampling-Regel definiert sich anhand ihres Namens. Im Gegensatz zu Gruppen ARNs haben Stichprobenregeln keine eindeutig generierte ID.

Fehlerbehebung bei AWS X-Ray Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit X-Ray und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion in X-Ray durchzuführen
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich bin Administrator und möchte anderen den Zugriff auf X-Ray ermöglichen
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine X-Ray-Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion in X-Ray durchzuführen

Wenn Ihnen AWS-Managementkonsole mitgeteilt wird, dass Sie nicht berechtigt sind, eine Aktion auszuführen, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` Benutzer versucht, die Konsole zu verwenden, um Details zu einer Stichprobenregel anzuzeigen, aber nicht über die `xray:GetSamplingRules` entsprechenden Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: xray:GetSamplingRules on resource: arn:${Partition}:xray:${Region}:
${Account}:sampling-rule/${SamplingRuleName}
```

In diesem Fall bittet Mateo seinen Administrator, seine Richtlinien zu aktualisieren, um ihm den Zugriff auf die Sampling-Regel-Ressource mit der `xray:GetSamplingRules`-Aktion zu ermöglichen.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion auszuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an X-Ray übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in X-Ray auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin Administrator und möchte anderen den Zugriff auf X-Ray ermöglichen

Um anderen den Zugriff auf X-Ray zu ermöglichen, müssen Sie den Personen oder Anwendungen, die Zugriff benötigen, die Erlaubnis erteilen. Wenn Sie Benutzer und Anwendungen verwalten, weisen Sie Benutzern oder Gruppen Berechtigungssätze zu, um deren Zugriffsebene zu definieren. AWS IAM Identity Center Mit Berechtigungssätzen werden automatisch IAM-Richtlinien erstellt und den IAM-Rollen zugewiesen, die der Person oder Anwendung zugeordnet sind. Weitere Informationen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Berechtigungssätze.

Wenn Sie IAM Identity Center nicht verwenden, müssen Sie IAM-Entitäten (Benutzer oder Rollen) für die Personen oder Anwendungen erstellen, die Zugriff benötigen. Anschließend müssen Sie der Entität eine Richtlinie hinzufügen, die ihr die richtigen Berechtigungen in X-Ray gewährt. Nachdem die Berechtigungen erteilt wurden, geben Sie die Anmeldeinformationen an den Benutzer oder Anwendungsentwickler weiter. Sie werden diese Anmeldeinformationen für den Zugriff verwenden

AWS. Weitere Informationen zum Erstellen von IAM-Benutzern, -Gruppen, -Richtlinien und -Berechtigungen finden Sie im IAM-Benutzerhandbuch unter IAM-Identitäten sowie Richtlinien und Berechtigungen in IAM.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine X-Ray-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob X-Ray diese Funktionen unterstützt, finden Sie unterWie AWS X-Ray funktioniert mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Einloggen und Überwachen AWS X-Ray

Die Überwachung ist ein wichtiger Teil der Aufrechterhaltung von Zuverlässigkeit, Verfügbarkeit und Performance Ihrer AWS -Lösungen. Sie sollten Überwachungsdaten aus allen Teilen Ihrer AWS Lösung sammeln, damit Sie einen etwaigen Ausfall an mehreren Stellen leichter debuggen können. AWS bietet mehrere Tools zur Überwachung Ihrer X-Ray-Ressourcen und zur Reaktion auf potenzielle Vorfälle:

AWS CloudTrail Logs

AWS X-Ray lässt sich integrieren AWS CloudTrail , um API-Aktionen aufzuzeichnen, die von einem Benutzer, einer Rolle oder einem AWS Dienst in X-Ray ausgeführt wurden. Sie können CloudTrail damit X-Ray-API-Anfragen in Echtzeit überwachen und Protokolle in Amazon S3, Amazon CloudWatch Logs und Amazon CloudWatch Events speichern. Weitere Informationen finden Sie unter Protokollierung von X-Ray-API-Aufrufen mit AWS CloudTrail.

AWS Config Nachverfolgung

AWS X-Ray integriert sich in AWS Config , um Konfigurationsänderungen aufzuzeichnen, die an Ihren X-Ray-Verschlüsselungsressourcen vorgenommen wurden. Sie können AWS Config damit Ressourcen für die X-Ray-Verschlüsselung inventarisieren, den X-Ray-Konfigurationsverlauf überprüfen und Benachrichtigungen auf Grundlage von Ressourcenänderungen versenden. Weitere Informationen finden Sie unter Verfolgen von Konfigurationsänderungen bei der X-Ray-Verschlüsselung mit AWS Config.

CloudWatch Amazon-Überwachung

Sie können das X-Ray SDK for Java verwenden, um CloudWatch Amazon-Metriken ohne Stichproben aus Ihren gesammelten X-Ray-Segmenten zu veröffentlichen. Diese Metriken werden von der Start- und Endzeit des Segments sowie den Status-Flags für Fehler, Ausfall und Ablehnung abgeleitet. Mit diesen Trace-Metriken können Sie Wiederholungen und Abhängigkeitsprobleme in Teilsegmenten anzeigen. Weitere Informationen finden Sie unter AWS X-Ray Metriken für das X-Ray SDK for Java.

Konformitätsvalidierung für AWS X-Ray

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Resilienz in AWS X-Ray

Die AWS globale Infrastruktur basiert auf Availability AWS-Regionen Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Infrastruktursicherheit in AWS X-Ray

Als verwalteter Dienst AWS X-Ray ist er durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf X-Ray zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Verwendung AWS X-Ray mit VPC-Endpunkten

Wenn Sie Amazon Virtual Private Cloud (Amazon VPC) zum Hosten Ihrer AWS Ressourcen verwenden, können Sie eine private Verbindung zwischen Ihrer VPC und X-Ray herstellen. Dadurch können Ressourcen in Ihrer Amazon VPC mit dem X-Ray-Service kommunizieren, ohne das öffentliche Internet nutzen zu müssen.

Amazon VPC ist eine AWS-Service , mit der Sie AWS Ressourcen in einem von Ihnen definierten virtuellen Netzwerk starten können. Mit einer VPC haben Sie die Kontrolle über Ihre

Netzwerkeinstellungen, wie IP-Adressbereich, Subnetze, Routing-Tabellen und Netzwerk-Gateways. Um Ihre VPC mit X-Ray zu verbinden, definieren Sie einen VPC-Schnittstellen-Endpunkt. Der Endpunkt bietet zuverlässige, skalierbare Konnektivität zu X-Ray, ohne dass ein Internet-Gateway, eine NAT-Instanz (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen finden Sie unter Was ist Amazon VPC im Benutzerhandbuch zu Amazon VPC.

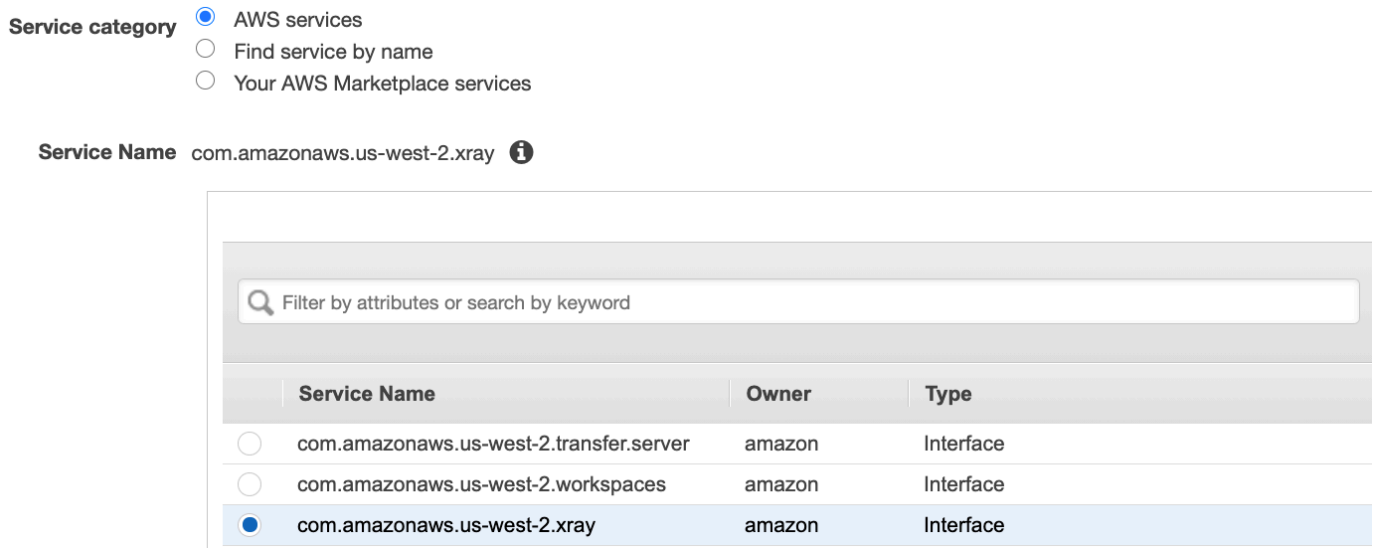
Schnittstelle, auf der VPC-Endpunkte basieren AWS PrivateLink, eine AWS Technologie, die eine private Kommunikation zwischen Benutzern AWS-Services mithilfe einer elastic network interface mit privaten IP-Adressen ermöglicht. Weitere Informationen finden Sie im AWS-Services Blogbeitrag „Neu — AWS PrivateLink für“ und „Erste Schritte“ im Amazon VPC-Benutzerhandbuch.

Um sicherzustellen, dass Sie in der von Ihnen ausgewählten Umgebung einen VPC-Endpunkt für X-Ray erstellen können AWS-Region, finden Sie unterUnterstützte Regionen.

Einen VPC-Endpunkt für X-Ray erstellen

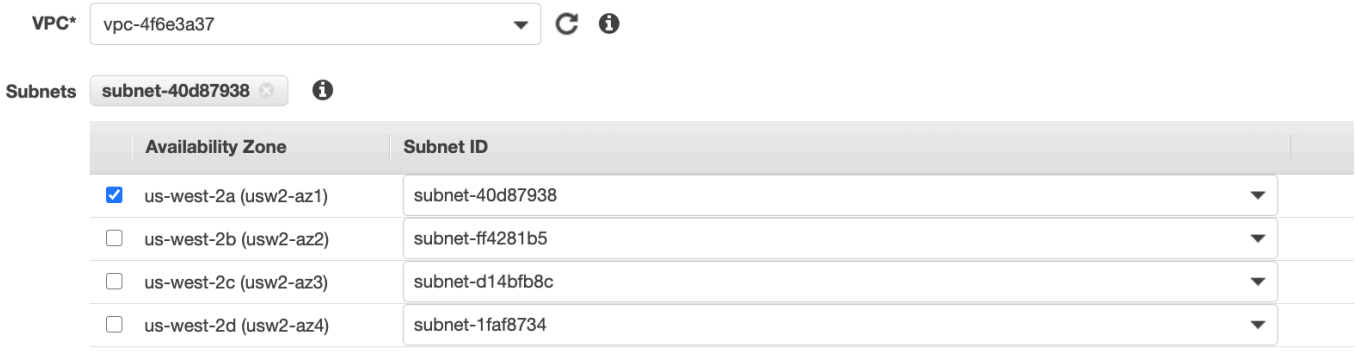
Um X-Ray mit Ihrer VPC zu verwenden, erstellen Sie einen VPC-Schnittstellen-Endpunkt für X-Ray.

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Navigieren Sie im Navigationsbereich zu Endpoints und wählen Sie Create Endpoint aus.
3. Suchen Sie nach dem Namen des AWS X-Ray Dienstes und wählen Sie ihn aus:`com.amazonaws.region.xray`.

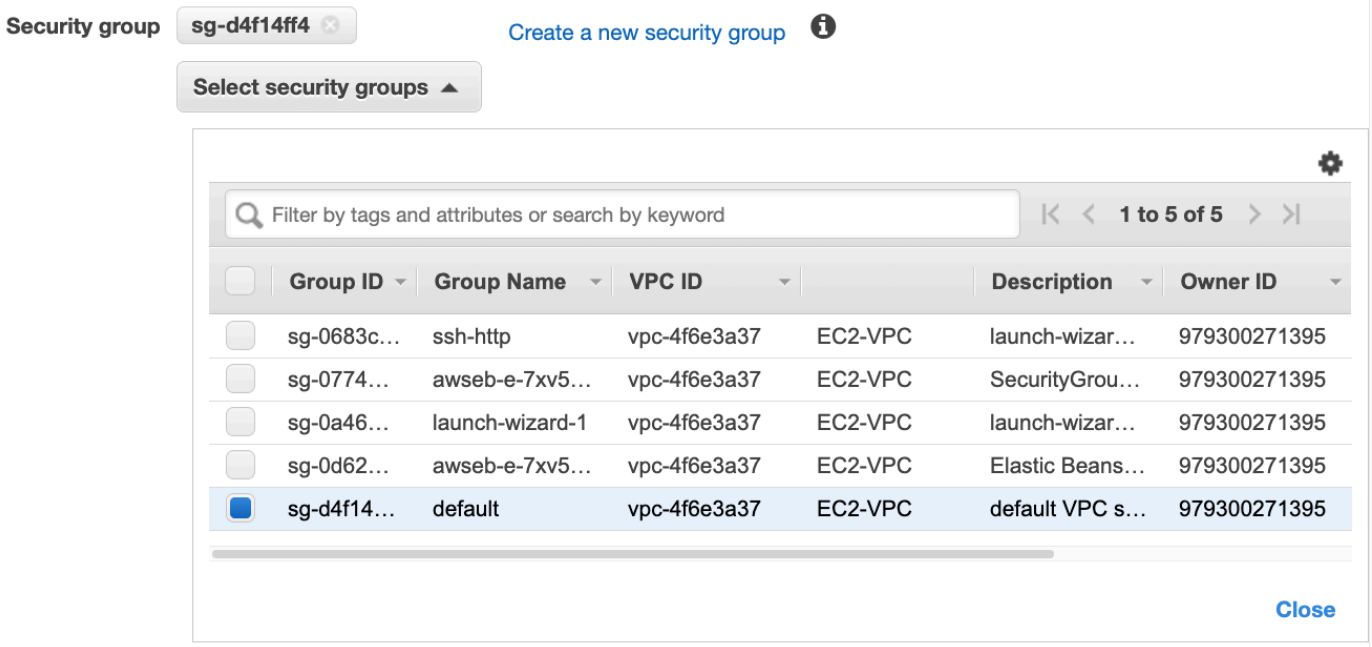


4. Wählen Sie die gewünschte VPC und dann ein Subnetz in Ihrer VPC aus, um den Schnittstellenendpunkt zu verwenden. Im ausgewählten Subnetz wird eine Endpunkt-Netzwerkschnittstelle erstellt. Sie können mehrere Subnetze in unterschiedlichen

Availability Zones angeben (wie vom Service unterstützt), um sicherzustellen, dass Ihr Schnittstellenendpunkt robust gegenüber Ausfällen von Availability Zone ist. Wenn Sie dies tun, wird in jedem von Ihnen angegebenen Subnetz eine Schnittstellen-Netzwerkschnittstelle erstellt.



5. (Optional) Private DNS ist standardmäßig für den Endpunkt aktiviert, sodass Sie Anfragen an X-Ray mit seinem Standard-DNS-Hostnamen stellen können. Sie können sich dafür entscheiden, es zu deaktivieren.
6. Geben Sie die Sicherheitsgruppen an, die der Endpunktnetzwerkschnittstelle zugeordnet werden sollen.



7. (Optional) Geben Sie eine benutzerdefinierte Richtlinie an, um die Zugriffsberechtigungen für den X-Ray-Dienst zu steuern. Standardmäßig ist Vollzugriff erlaubt.

Steuern des Zugriffs auf Ihren X-Ray-VPC-Endpunkt

Eine VPC-Endpunktrichtlinie ist eine IAM-Ressourcenrichtlinie, die Sie einem Endpunkt beim Erstellen oder Ändern des Endpunkts zuordnen. Wenn Sie einem Endpunkt beim Erstellen keine Richtlinie zuordnen, ordnet Amazon VPC ihm eine Standardrichtlinie mit Vollzugriff auf den Service zu. IAM-Benutzerrichtlinien oder servicespezifische Richtlinien werden von einer Endpunktrichtlinie nicht überschrieben oder ersetzt. Endpunktrichtlinien steuern unabhängig vom Endpunkt den Zugriff auf den angegebenen Service. Endpunktrichtlinien müssen im JSON-Format erstellt werden. Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon VPC User Guide.

Mit der VPC-Endpunktrichtlinie können Sie die Berechtigungen für verschiedene X-Ray-Aktionen steuern. Sie können beispielsweise eine Richtlinie erstellen, um nur andere Aktionen zuzulassen PutTraceSegment und alle anderen Aktionen abzulehnen. Dadurch werden Workloads und Dienste in der VPC darauf beschränkt, nur Trace-Daten an X-Ray zu senden und alle anderen Aktionen wie das Abrufen von Daten, das Ändern der Verschlüsselungskonfiguration oder das Erstellen/Aktualisieren von Gruppen zu verweigern.

Das Folgende ist ein Beispiel für eine Endpunktrichtlinie für X-Ray. Diese Richtlinie ermöglicht Benutzern, die über die VPC eine Verbindung zu X-Ray herstellen, Segmentdaten an X-Ray zu senden, und verhindert außerdem, dass sie andere X-Ray-Aktionen ausführen.

```
{"Statement": [
    {"Sid": "Allow PutTraceSegments",
      "Principal": "*",
      "Action": [
        "xray:PutTraceSegments"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

So bearbeiten Sie die VPC-Endpunktrichtlinie für X-Ray

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsbereich Endpunkte aus.

3. Wenn Sie den Endpunkt für X-Ray noch nicht erstellt haben, folgen Sie den Schritten unterEinen VPC-Endpunkt für X-Ray erstellen.
4. Wählen Sie com.amazonaws aus. *region*.xray-Endpunkt und wählen Sie dann die Registerkarte Richtlinie aus.
5. Klicken Sie auf Edit Policy (Richtlinie bearbeiten) und nehmen Sie Ihre Änderungen vor.

Unterstützte Regionen

X-Ray unterstützt derzeit VPC-Endpunkte in den folgenden Bereichen: AWS-Regionen

- US East (Ohio)
- USA Ost (Nord-Virginia)
- USA West (Nordkalifornien)
- USA West (Oregon)
- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asien-Pazifik (Osaka)
- Asien-Pazifik (Seoul)
- Asien-Pazifik (Singapur)
- Asien-Pazifik (Sydney)
- Asien-Pazifik (Tokio)
- Canada (Central)
- Europe (Frankfurt)
- Europa (Irland)
- Europa (London)
- Europa (Milan)
- Europe (Paris)
- Europe (Stockholm)
- Middle East (Bahrain)
- Südamerika (São Paulo)
- AWS GovCloud (US-Ost)

- AWS GovCloud (US-West)

Serviceübergreifende Confused-Deputy-Prävention

Das Confused-Deputy-Problem ist ein Sicherheitsproblem, bei dem eine juristische Stelle, die nicht über die Berechtigung zum Ausführen einer Aktion verfügt, eine privilegiertere juristische Stelle zwingen kann, die Aktion auszuführen. In AWS, dienstübergreifender Identitätswechsel kann zum Problem des verwirrten Stellvertreters führen. Ein dienstübergreifender Identitätswechsel kann auftreten, wenn ein Dienst (der Anruf-Dienst) einen anderen Dienst anruft (den aufgerufenen Dienst). Der aufrufende Service kann manipuliert werden, um seine Berechtigungen zu verwenden, um Aktionen auf die Ressourcen eines anderen Kunden auszuführen, für die er sonst keine Zugriffsberechtigung haben sollte. Um dies zu verhindern, bietet AWS Tools, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben.

Wir empfehlen `aws:SourceArn`, die Kontextschlüssel, `aws:SourceAccount``aws:SourceOrgID`, und die `aws:SourceOrgPaths`globalen Bedingungsschlüssel in Ressourcenrichtlinien zu verwenden, um die Berechtigungen einzuschränken, die xraylong der Ressource einem anderen Dienst erteilt. Verwenden Sie `aws:SourceArn`, um nur einer Ressource den serviceübergreifenden Zugriff zuzuordnen. Verwenden Sie `aws:SourceAccount`, um jede Ressource in diesem Konto der serviceübergreifenden Nutzung zuzuordnen. Verwenden Sie `aws:SourceOrgID`, damit alle Ressourcen von allen Konten innerhalb einer Organisation der serviceübergreifenden Nutzung zugeordnet werden können. Verwenden Sie `aws:SourceOrgPaths`, um beliebige Ressourcen von Konten innerhalb eines AWS Organizations -Pfades der serviceübergreifenden Nutzung zuzuordnen. Weitere Informationen zur Verwendung und zum Verständnis von Pfaden finden Sie unter Grundlegendes zum AWS Organizations Entitätspfad.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den globalen Kontextbedingungsschlüssel `aws:SourceArn` mit Platzhalterzeichen (`*`) für die unbekannten Teile des ARN. Beispiel, `arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`.

Wenn der `aws:SourceArn`-Wert die Konto-ID nicht enthält, z. B. einen Amazon-S3-Bucket-ARN, müssen Sie sowohl `aws:SourceAccount` als auch `aws:SourceArn` verwenden, um Berechtigungen einzuschränken.

Um sich vor dem Confused-Deputy-Problem im großen Maßstab zu schützen, verwenden Sie den globalen Bedingungskontextschlüssel `aws:SourceOrgID` oder `aws:SourceOrgPaths` mit der Organisations-ID oder dem Organisationspfad der Ressource in Ihren ressourcenbasierten Richtlinien. Richtlinien, die den Schlüssel `aws:SourceOrgID` oder `aws:SourceOrgPaths` enthalten, schließen automatisch die richtigen Konten ein und Sie müssen die Richtlinien nicht manuell aktualisieren, wenn Sie Konten in Ihrer Organisation hinzufügen, entfernen oder verschieben.

Das folgende Beispiel zeigt, wie Sie die Kontextschlüssel `aws:SourceArn` und die `aws:SourceAccount` globalen Bedingungsschlüssel in Xray verwenden können, um das Problem des verwirrten Stellvertreters zu vermeiden.

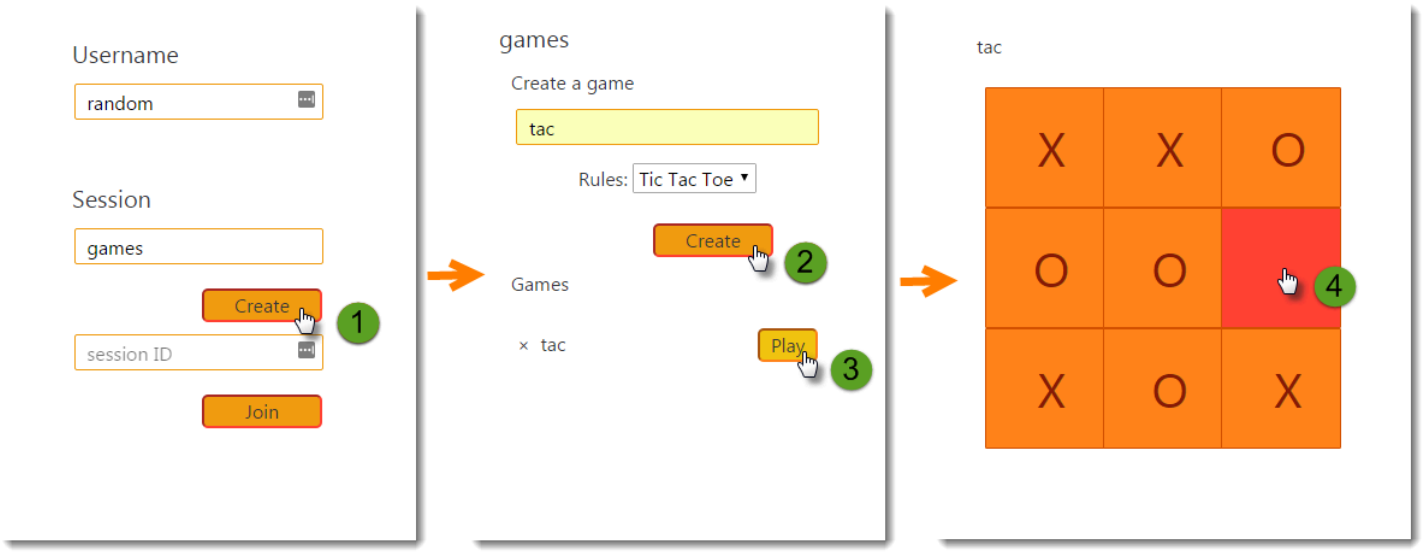
```
{
      "Sid": "BlockCrossAccountUnlessSameSource",
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKeyWithoutPlaintext"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": "123456789012",
          "aws:SourceAccount": "123456789012"
        },
        "ArnNotLike": {
          "aws:SourceArn": "arn:*:*:*:123456789012:*"
        }
      }
    }
```

AWS X-Ray Musteranwendung

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

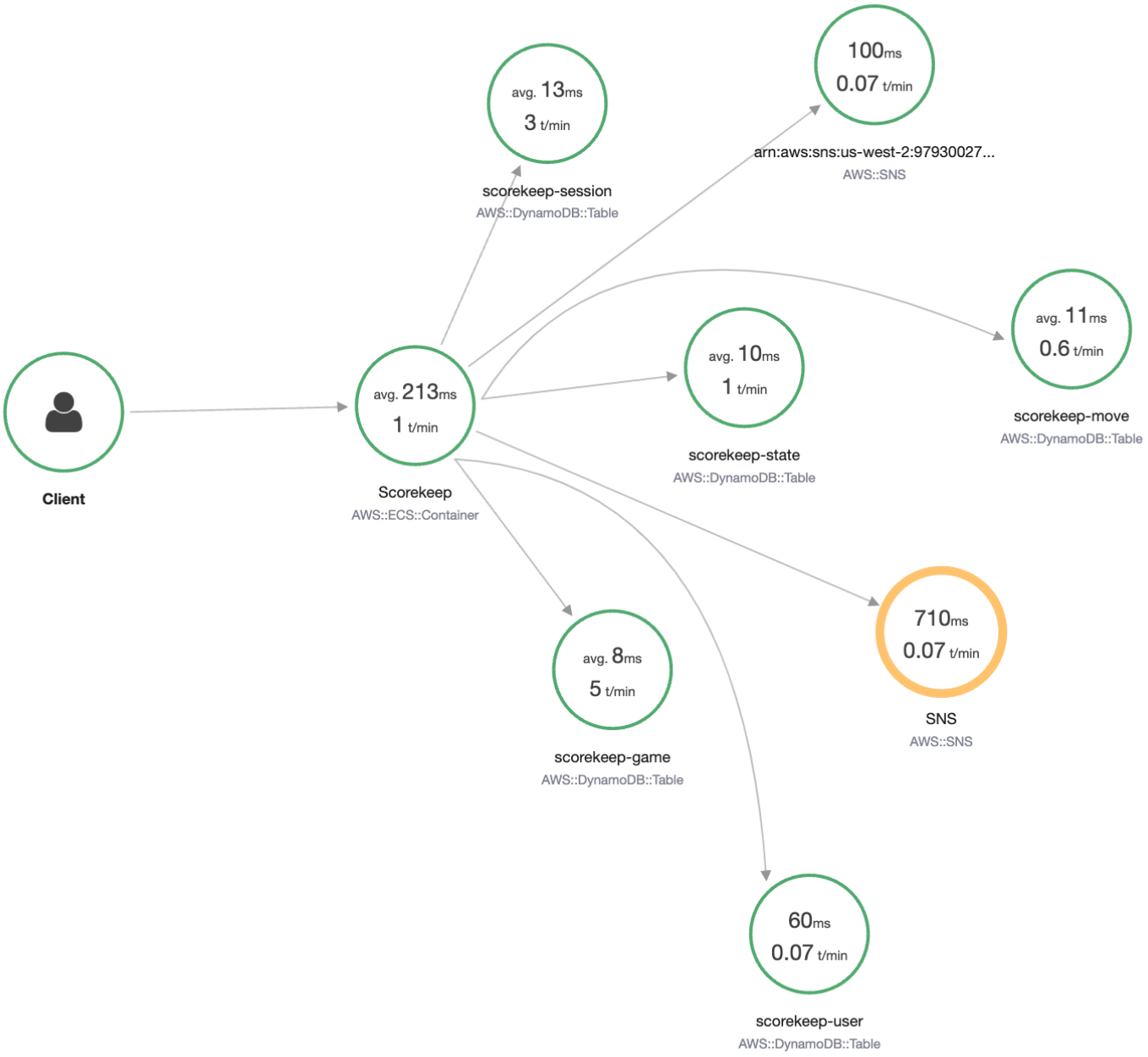
Die AWS eb-java-scorekeepX-Ray-Beispiel-App, verfügbar auf GitHub, zeigt die Verwendung des AWS X-Ray-SDK zur Instrumentierung eingehender HTTP-Aufrufe, DynamoDB-SDK-Clients und HTTP-Clients. Die Beispiel-App wird verwendet, CloudFormation um DynamoDB-Tabellen zu erstellen, Java-Code auf einer Instanz zu kompilieren und den X-Ray-Daemon ohne zusätzliche Konfiguration auszuführen.

Sehen Sie sich das Scorekeep-Tutorial an, um mit der Installation und Verwendung einer instrumentierten Beispielanwendung zu beginnen. Verwenden Sie dazu das oder das. AWS-Managementkonsole AWS CLI



Das Beispiel umfasst eine Front-End-Web-App, die API, die sie aufruft, und die DynamoDB-Tabellen, die sie zum Speichern von Daten verwendet. Die grundlegende Instrumentierung mit Filtern, Plugins

und instrumentierten AWS SDK-Clients wird im Zweig des Projekts gezeigt. `xray-gettingstarted` Dies ist die Verzweigung, die Sie im Tutorial "Erste Schritte" bereitstellen. Da diese Verzweigung nur die Grundlagen beinhaltet, können Sie einen diff-Vorgang mit der `master`-Verzweigung durchführen, um schnell die Grundlagen zu erfassen.



Die Beispielanwendung veranschaulicht die grundlegende Instrumentierung in folgenden Dateien:

- HTTP-Anforderungsfilter — `WebConfig.java`
- AWS Instrumentierung des SDK-Clients — `build.gradle`

Der `xray` Zweig der Anwendung umfasst die Verwendung von Anmerkungen HTTPClient, SQL-Abfragen, benutzerdefinierten Untersegmenten, einer instrumentierten AWS LambdaFunktion sowie instrumentiertem Initialisierungscode und Skripten.

Um die Benutzeranmeldung und die AWS SDK für JavaScript Nutzung im Browser zu unterstützen, fügt die `xray-cognito` Filiale Amazon Cognito hinzu, um die Benutzerauthentifizierung und -autorisierung zu unterstützen. Mit den von Amazon Cognito abgerufenen Anmeldeinformationen sendet die Web-App auch Trace-Daten an X-Ray, um Anforderungsinformationen aus Kundensicht aufzuzeichnen. Der Browser-Client erscheint als eigener Knoten auf der Trace-Map und zeichnet zusätzliche Informationen auf, darunter die URL der Seite, die der Benutzer gerade betrachtet, und die Benutzer-ID.

Schließlich fügt der `xray-worker` Branch eine instrumentierte Python-Lambda-Funktion hinzu, die unabhängig ausgeführt wird und Elemente aus einer Amazon SQS SQS-Warteschlange verarbeitet. Scorekeep fügt ein Element zur Warteschlange hinzu, wenn ein Spiel endet. Der Lambda-Worker, ausgelöst durch CloudWatch Ereignisse, ruft alle paar Minuten Elemente aus der Warteschlange ab und verarbeitet sie, um Spielaufzeichnungen zur Analyse in Amazon S3 zu speichern.

Topics

- Erste Schritte mit der Scorekeep-Beispielanwendung
- Manuelles Instrumentieren von AWS SDK-Clients
- Erstellen zusätzlicher Untersegmente
- Anmerkungen, Metadaten und Benutzer aufzeichnen IDs
- Instrumentieren von ausgehenden HTTP-Aufrufen
- Instrumentieren von Aufrufen einer PostgreSQL-Datenbank
- Instrumentierungsfunktionen AWS Lambda
- Instrumentieren von Startup-Code
- Instrumentieren von Skripten
- Instrumentieren eines Web-App-Clients
- Verwenden instrumentierter Clients in Auftragnehmer-Threads

Erste Schritte mit der Scorekeep-Beispielanwendung

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

In diesem Tutorial wird der `xray-gettingstarted` Zweig der Scorekeep-Beispielanwendung verwendet CloudFormation , mit dem die Ressourcen erstellt und konfiguriert werden, mit denen die Beispielanwendung und der X-Ray-Daemon auf Amazon ECS ausgeführt werden. Die Anwendung verwendet das Spring-Framework, um eine JSON-Web-API AWS SDK für Java zu implementieren und Daten in Amazon DynamoDB zu speichern. Ein Servlet-Filter in der Anwendung instrumentiert alle eingehenden Anfragen, die von der Anwendung bedient werden, und ein Anforderungshandler auf dem AWS SDK-Client instrumentiert Downstream-Aufrufe an DynamoDB.

Sie können diesem Tutorial entweder mit dem oder dem AWS-Managementkonsole folgen. AWS CLI

Sections

Voraussetzungen

In diesem Tutorial werden CloudFormation die Ressourcen erstellt und konfiguriert, auf denen die Beispielanwendung und der X-Ray-Daemon ausgeführt werden. Für die Installation und Ausführung des Tutorials sind die folgenden Voraussetzungen erforderlich:

1. Wenn Sie einen IAM-Benutzer mit eingeschränkten Rechten verwenden, fügen Sie die folgenden Benutzerrichtlinien in der IAM-Konsole hinzu:

   - `AWSCloudFormationFullAccess`— für den Zugriff und die Nutzung CloudFormation
   - `AmazonS3FullAccess`— um eine Vorlagendatei hochzuladen, CloudFormation um die AWS-Managementkonsole
   - `IAMFullAccess`— um die Amazon ECS- und EC2 Amazon-Instance-Rollen zu erstellen
   - `AmazonEC2FullAccess`— um die EC2 Amazon-Ressourcen zu erstellen
   - `AmazonDynamoDBFullAccess`— um die DynamoDB-Tabellen zu erstellen
   - `AmazonECS_FullAccess`— um Amazon ECS-Ressourcen zu erstellen
   - `AmazonSNSFullAccess`— um das Amazon SNS SNS-Thema zu erstellen
   - `AWSXrayReadOnlyAccess`— für die Erlaubnis, die Trace-Map und die Traces in der X-Ray-Konsole anzusehen
2. Um das Tutorial mit dem auszuführen AWS CLI, installieren Sie die CLI-Version 2.7.9 oder höher und konfigurieren Sie die CLI mit dem Benutzer aus dem vorherigen Schritt. Stellen Sie sicher, dass die Region konfiguriert ist, wenn Sie das AWS CLI mit dem Benutzer konfigurieren. Wenn eine Region nicht konfiguriert ist, müssen Sie sie `--region` *`AWS-REGION`* an jeden CLI-Befehl anhängen.
3. Stellen Sie sicher, dass Git installiert ist, um das Beispielanwendungs-Repo zu klonen.
4. Verwenden Sie das folgende Codebeispiel, um den `xray-gettingstarted` Zweig des Scorekeep-Repositorys zu klonen:

   ```
   git clone https://github.com/aws-samples/eb-java-scorekeep.git xray-scorekeep -b
    xray-gettingstarted
   ```

Installieren Sie die Scorekeep-Anwendung mit CloudFormation

AWS-Managementkonsole

Installieren Sie die Beispielanwendung mit dem AWS-Managementkonsole

1. Öffnen Sie die CloudFormation-Konsole.
2. Wählen Sie Stack erstellen und dann im Drop-down-Menü die Option Mit neuen Ressourcen aus.
3. Wählen Sie im Abschnitt Specify template (Vorlage angeben) die Option Upload a template file (Vorlagendatei hochladen) aus.
4. Wähle „Datei auswählen“, navigiere zu dem `xray-scorekeep/cloudformation` Ordner, der beim Klonen des Git-Repositorys erstellt wurde, und wähle die `cf-resources.yaml` Datei aus.
5. Wählen Sie Next (Weiter), um fortzufahren.
6. Geben Sie den Text `scorekeep` in das Textfeld Stack-Name ein und wählen Sie dann unten auf der Seite Weiter aus, um fortzufahren. Beachten Sie, dass im Rest dieses Tutorials davon ausgegangen wird, dass der Stack benannt `scorekeep` ist.
7. Scrollen Sie zum Ende der Seite „Stack-Optionen konfigurieren“ und wählen Sie Weiter, um fortzufahren.
8. Scrollen Sie zum Ende der Überprüfungsseite, aktivieren Sie das Kontrollkästchen zur Bestätigung, dass CloudFormation möglicherweise IAM-Ressourcen mit benutzerdefinierten Namen erstellt werden, und wählen Sie Stack erstellen aus.
9. Der CloudFormation Stack wird jetzt erstellt. Der Stack-Status wird `CREATE_IN_PROGRESS` etwa fünf Minuten lang angezeigt, bevor er zu wechselt`CREATE_COMPLETE`. Der Status wird regelmäßig aktualisiert, oder Sie können die Seite aktualisieren.

AWS CLI

Installieren Sie die Beispielanwendung mit dem AWS CLI

1. Navigieren Sie zu dem `cloudformation` Ordner des `xray-scorekeep` Repositorys, den Sie zuvor im Tutorial geklont haben:

```
cd xray-scorekeep/cloudformation/
```

2. Geben Sie den folgenden AWS CLI Befehl ein, um den CloudFormation Stack zu erstellen:

```
aws cloudformation create-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

3. Warten Sie`CREATE_COMPLETE`, bis der CloudFormation Stack-Status lautet. Dies dauert etwa fünf Minuten. Verwenden Sie den folgenden AWS CLI Befehl, um den Status zu überprüfen:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Generieren von Ablaufverfolgungsdaten

Die Beispielanwendung enthält eine Front-End-Webanwendung. Verwenden Sie die Web-App, um Traffic zur API zu generieren und Trace-Daten an X-Ray zu senden. Rufen Sie zunächst die URL der Web-App mit dem AWS-Managementkonsole oder dem ab AWS CLI:

AWS-Managementkonsole

Suchen Sie die Anwendungs-URL mithilfe von AWS-Managementkonsole

1. Öffnen Sie die CloudFormation-Konsole.
2. Wählen Sie den `scorekeep` Stapel aus der Liste aus.
3. Wählen Sie auf der `scorekeep` Stack-Seite den Tab Outputs und anschließend den `LoadBalancerUrl` URL-Link, um die Webanwendung zu öffnen.

AWS CLI

Suchen Sie die Anwendungs-URL mithilfe des AWS CLI

1. Verwenden Sie den folgenden Befehl, um die URL der Webanwendung anzuzeigen:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].Outputs[0].OutputValue"
```

2. Kopieren Sie diese URL und öffnen Sie sie in einem Browser, um die Scorekeep-Webanwendung anzuzeigen.

Verwenden Sie die Webanwendung, um Trace-Daten zu generieren

1. Wählen Sie Create aus, um einen Benutzer und eine Sitzung zu erstellen.
2. Geben Sie einen Game Name ein, legen Sie die Rules auf Tic Tac Toe fest und wählen Sie anschließend Create aus, um ein Spiel zu erstellen.
3. Wählen Sie Play, um das Spiel zu starten.
4. Wählen Sie eine Kachel, um einen Spielzug zu machen, und ändern Sie den Spielestatus.

Jeder dieser Schritte generiert HTTP-Anfragen an die API und Downstream-Aufrufe an DynamoDB, um Benutzer-, Sitzungs-, Spiel-, Bewegungs- und Statusdaten zu lesen und zu schreiben.

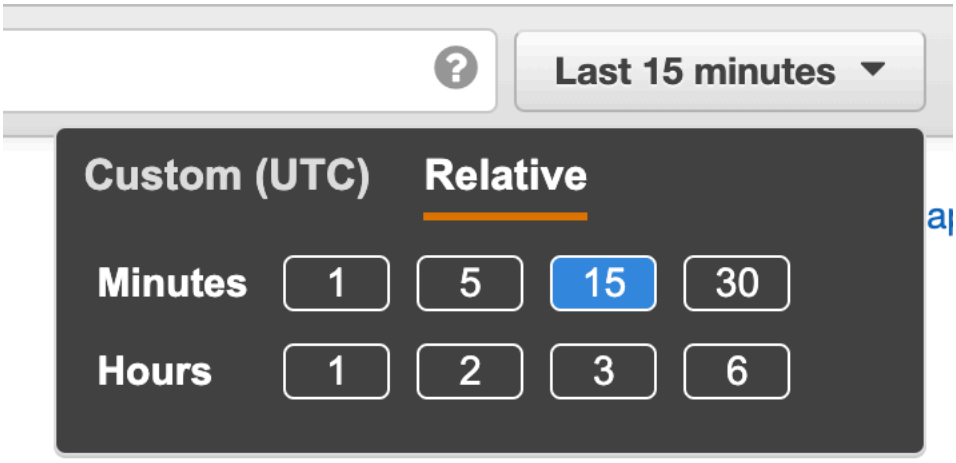
Sehen Sie sich die Trace-Map im AWS-Managementkonsole

Sie können die Trace-Map und die von der Beispielanwendung generierten Traces im X-Ray und in den CloudWatch Konsolen sehen.

X-Ray console

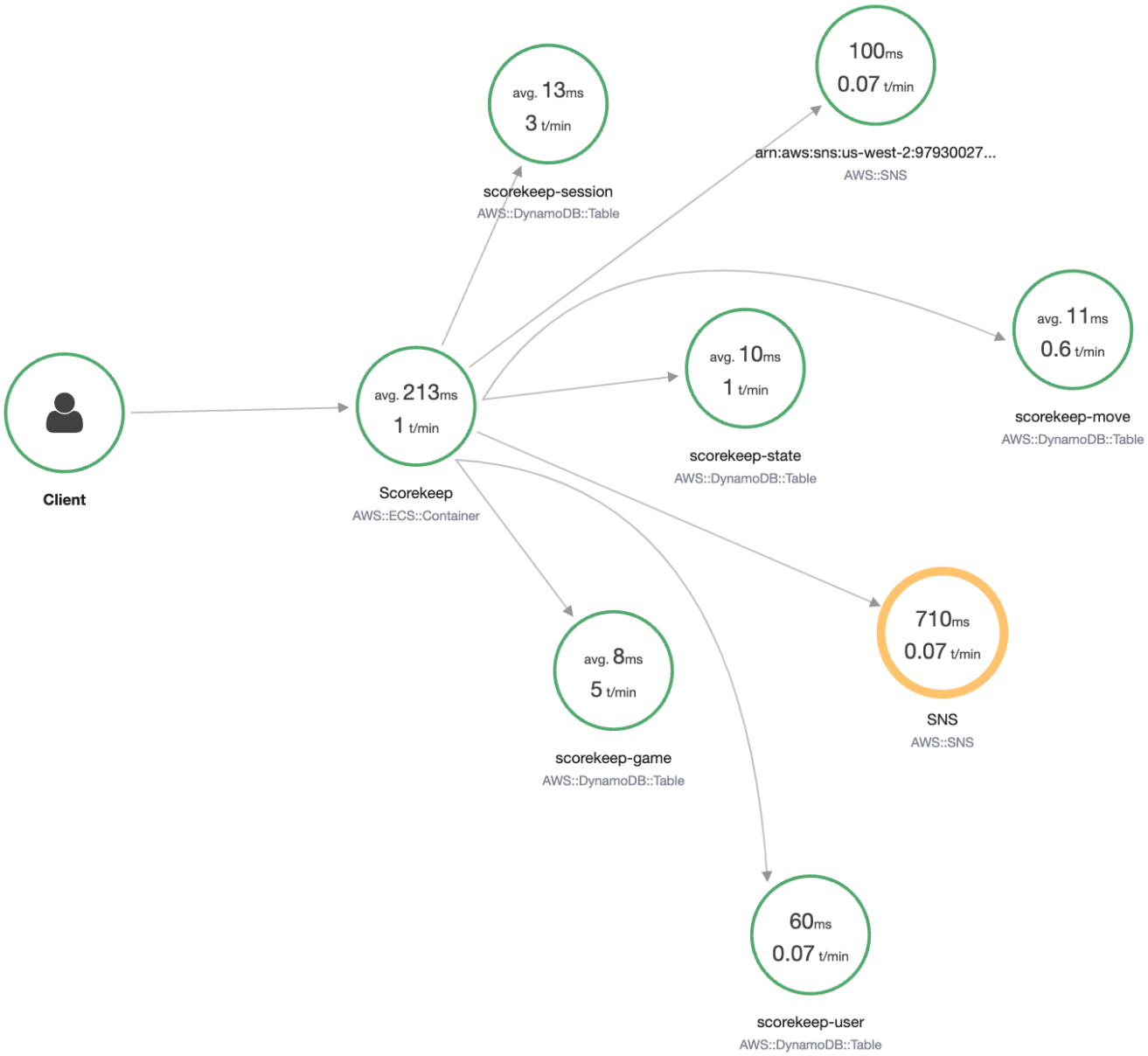
Verwenden Sie die X-Ray-Konsole

1. Öffnen Sie die Trace-Map-Seite der X-Ray-Konsole.
2. Die Konsole zeigt eine Darstellung des Service-Graphen, den X-Ray aus den von der Anwendung gesendeten Trace-Daten generiert. Achten Sie darauf, den Zeitraum der Trace-Map bei Bedarf anzupassen, um sicherzustellen, dass alle Traces seit dem ersten Start der Webanwendung angezeigt werden.



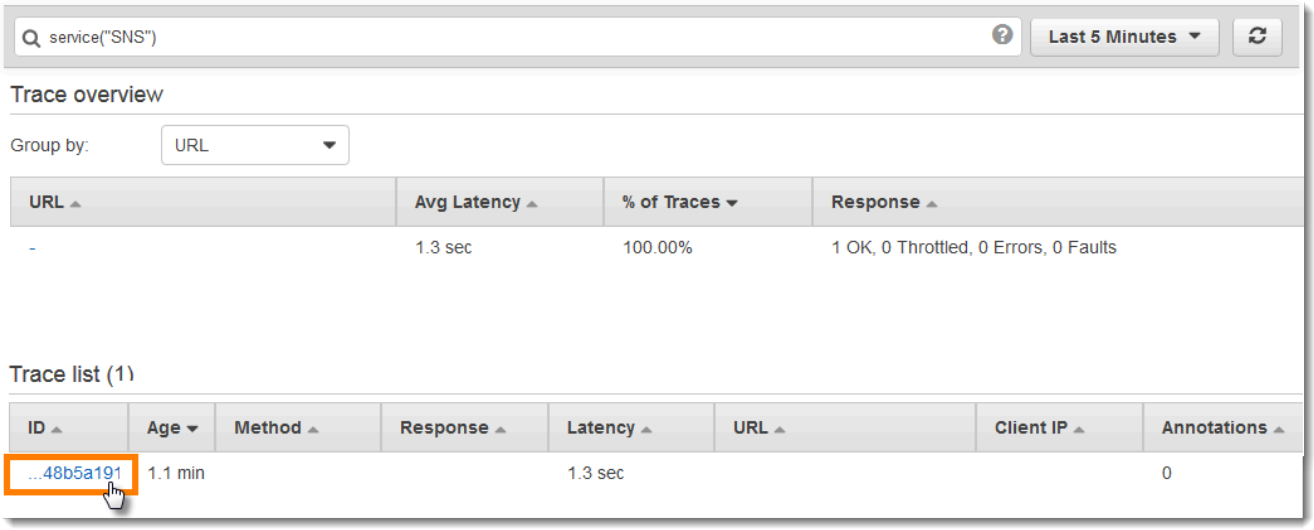
Die Trace-Map zeigt den Web-App-Client, die in Amazon ECS ausgeführte API und jede DynamoDB-Tabelle, die die Anwendung verwendet. Jede Anforderung an die Anwendung, bis zu einer konfigurierbaren Höchstanzahl von Anforderungen pro Sekunde, wird verfolgt, und zwar wie sie auf die API trifft, Anforderungen an nachgelagerte Services generiert und abgeschlossen wird.

Sie können jeden Knoten in der Service-Grafik wählen, um Ablaufverfolgungen für Anforderungen anzusehen, die Datenverkehr zu diesem Knoten generiert haben. Derzeit ist der Amazon SNS SNS-Knoten gelb. Zeigen Sie die Details an, um herauszufinden, warum.

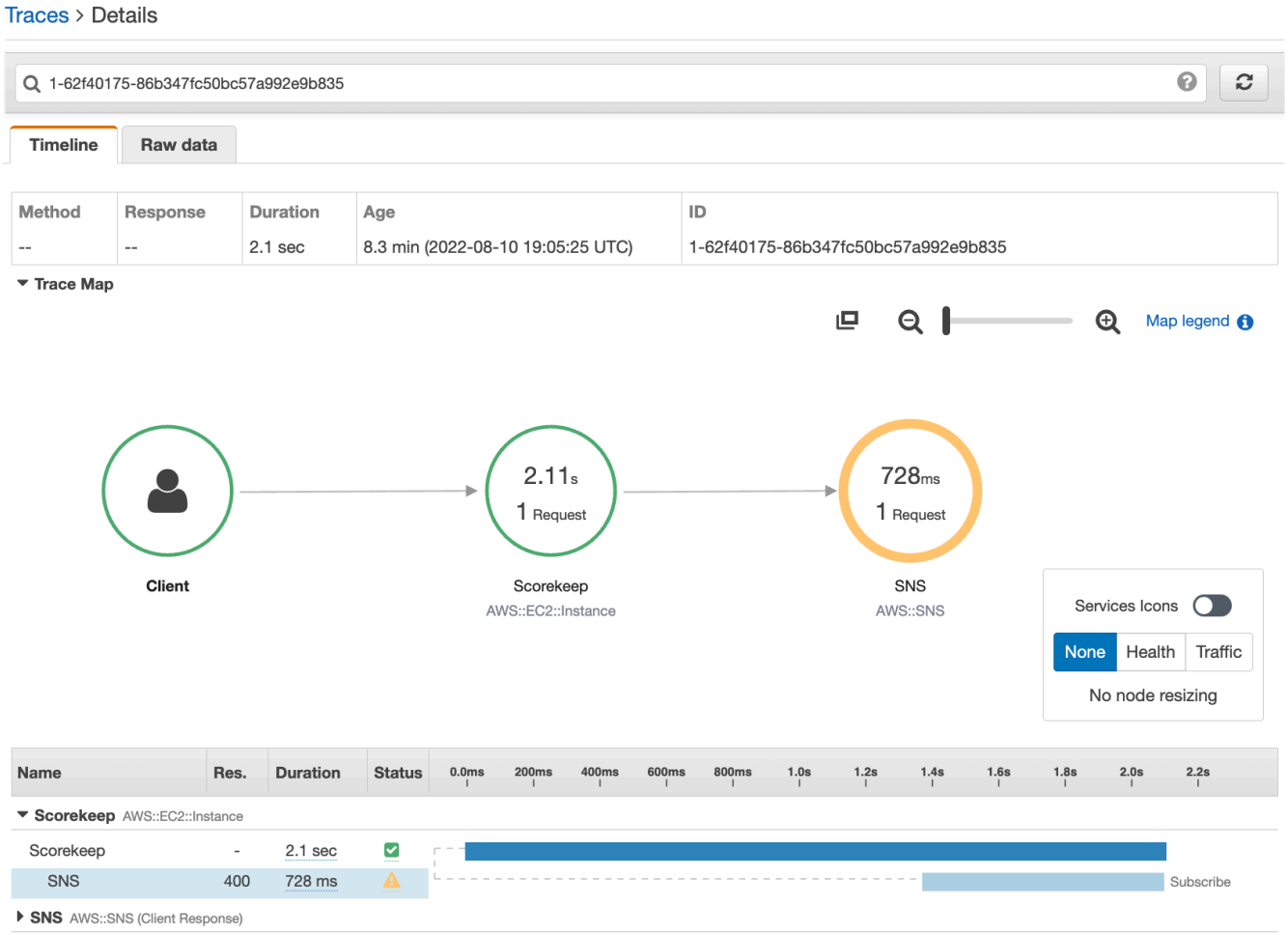


So finden Sie die Ursache des Fehlers

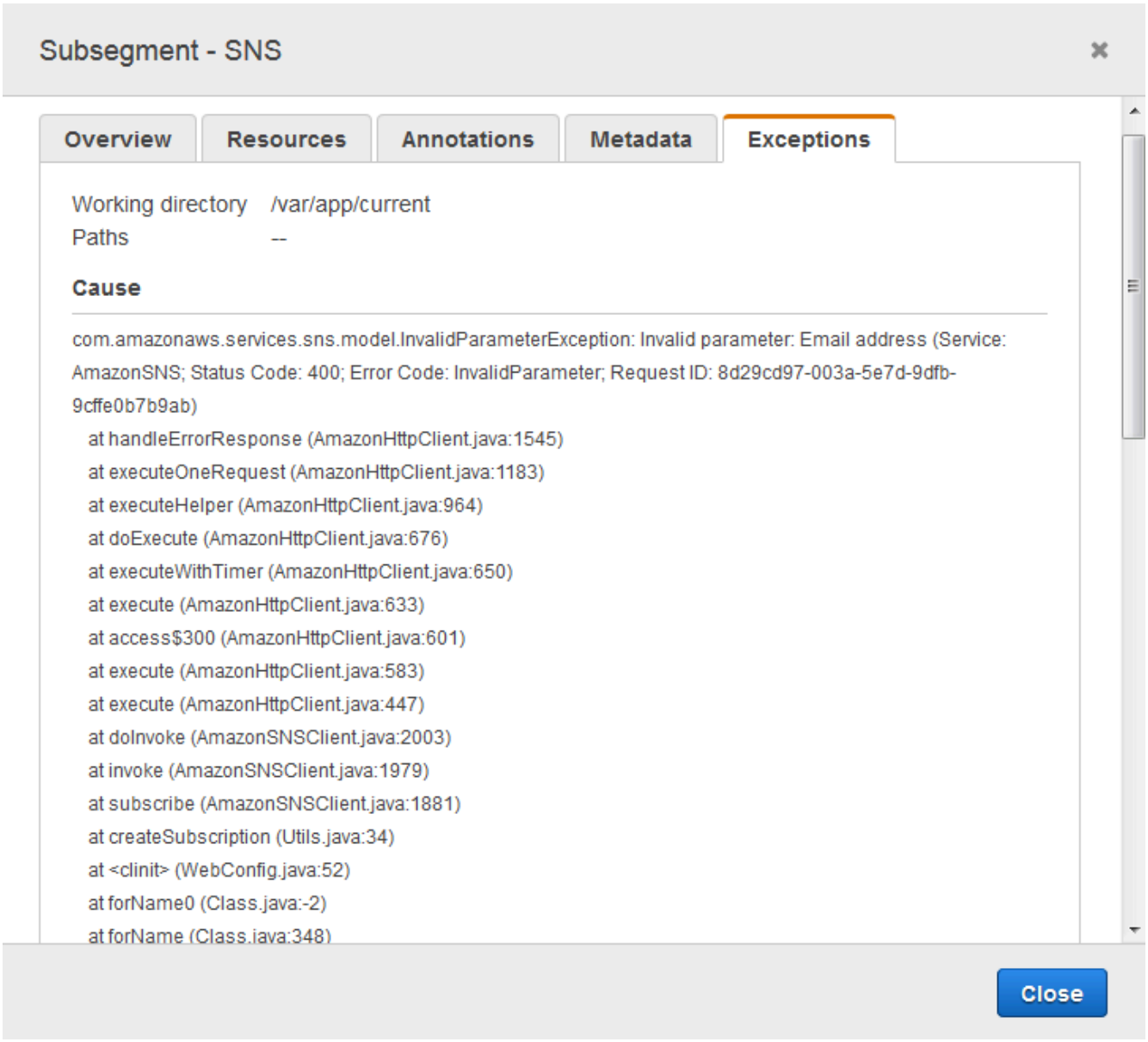
1. Wählen Sie den Knoten mit dem Namen SNS. Das Fenster mit den Knotendetails wird angezeigt.
2. Wählen Sie View traces (Ablaufverfolgungen anzeigen), um auf den Bildschirm Trace overview (Ablaufverfolgungsübersicht) zuzugreifen.
3. Wählen Sie die Nachverfolgung aus der Trace list. Diese Ablaufverfolgung verfügt über keine Methode oder URL, da sie während des Startups und nicht als Reaktion auf eine eingehende Anforderung aufgezeichnet wurde.



4. Wählen Sie das Fehlerstatussymbol im Amazon SNS SNS-Segment unten auf der Seite, um die Seite Ausnahmen für das SNS-Subsegment zu öffnen.



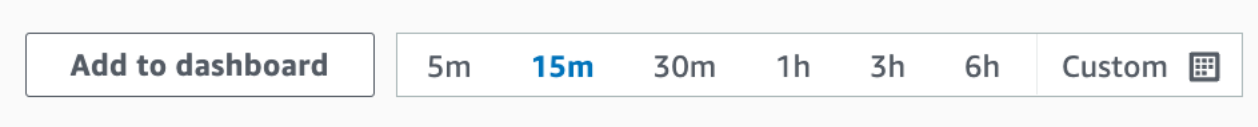
5. Das X-Ray SDK erfasst automatisch Ausnahmen, die von instrumentierten AWS SDK-Clients ausgelöst werden, und zeichnet den Stack-Trace auf.



CloudWatch console

Verwenden Sie die Konsole CloudWatch

1. Öffnen Sie die X-Ray-Trace-Map-Seite der CloudWatch Konsole.
2. Die Konsole zeigt eine Darstellung des Service-Graphen, den X-Ray aus den von der Anwendung gesendeten Trace-Daten generiert. Achten Sie darauf, den Zeitraum der Trace-Map bei Bedarf anzupassen, um sicherzustellen, dass alle Traces seit dem ersten Start der Webanwendung angezeigt werden.



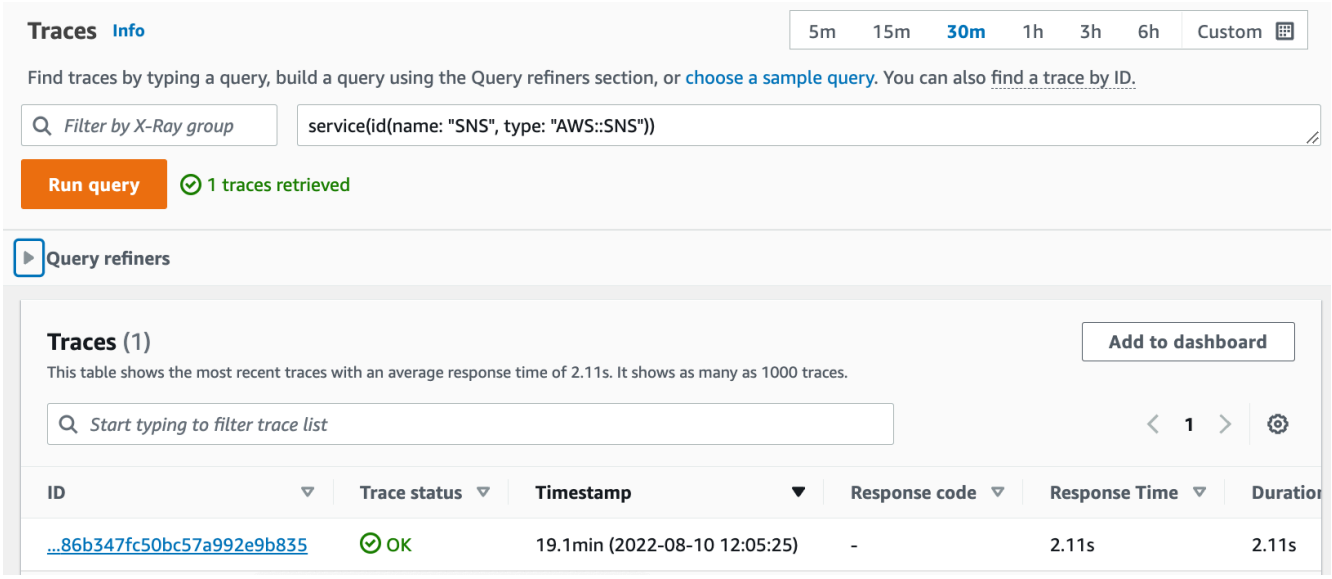
Die Trace-Map zeigt den Web-App-Client, die in Amazon EC2 ausgeführte API und jede DynamoDB-Tabelle, die die Anwendung verwendet. Jede Anforderung an die Anwendung, bis zu einer konfigurierbaren Höchstanzahl von Anforderungen pro Sekunde, wird verfolgt, und zwar wie sie auf die API trifft, Anforderungen an nachgelagerte Services generiert und abgeschlossen wird.

Sie können jeden Knoten in der Service-Grafik wählen, um Ablaufverfolgungen für Anforderungen anzusehen, die Datenverkehr zu diesem Knoten generiert haben. Derzeit ist der Amazon SNS SNS-Knoten orange. Zeigen Sie die Details an, um herauszufinden, warum.

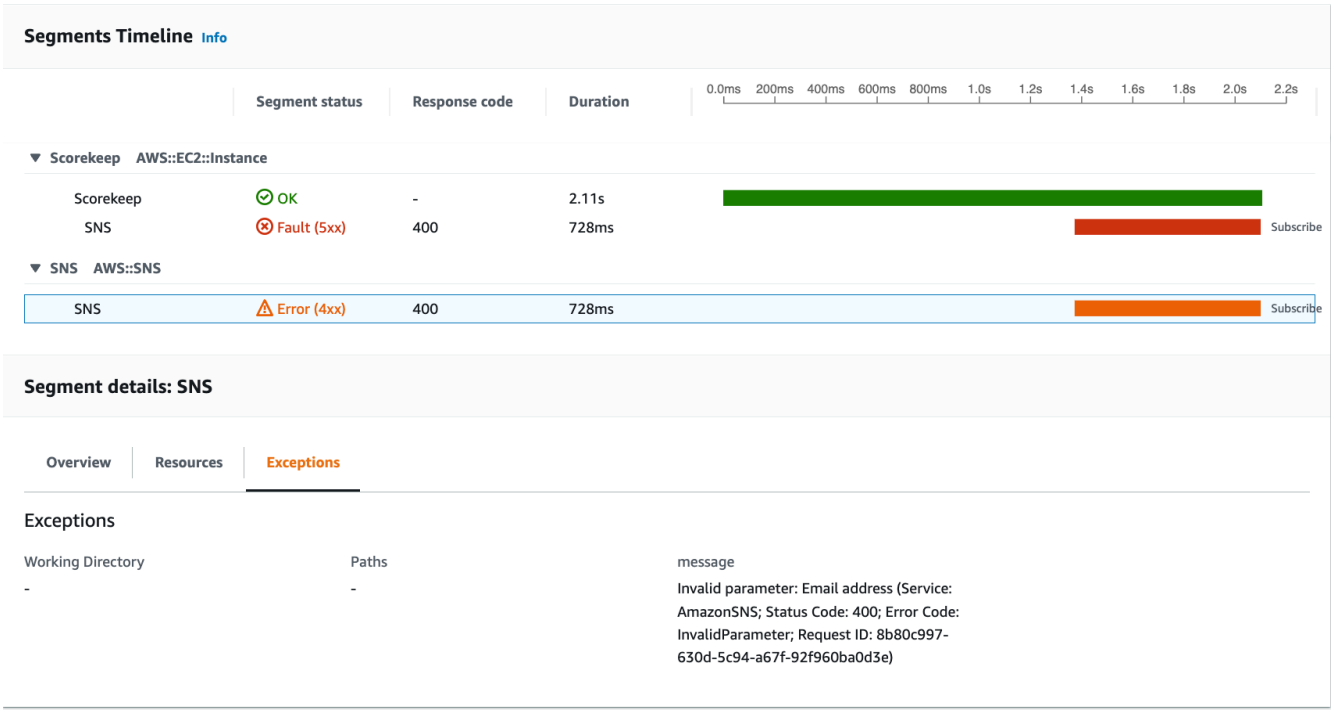

scorekeep-state
DynamoDB Table
scorekeep-game
DynamoDB Table
scorekeep-move
DynamoDB Table
Client
Scorekeep
ECS Container
SNS
SNS
scorekeep-session
DynamoDB Table

So finden Sie die Ursache des Fehlers

1. Wählen Sie den Knoten mit dem Namen SNS. Der Bereich mit den SNS-Knotendetails wird unter der Karte angezeigt.
2. Wählen Sie „Traces anzeigen“, um die Seite „Traces“ aufzurufen.
3. Fügen Sie das Ende der Seite hinzu und wählen Sie den Trace aus der Traces-Liste aus. Diese Ablaufverfolgung verfügt über keine Methode oder URL, da sie während des Startups und nicht als Reaktion auf eine eingehende Anforderung aufgezeichnet wurde.



4. Wählen Sie das Amazon SNS SNS-Untersegment unten in der Segment-Timeline aus und wählen Sie den Tab Ausnahmen für das SNS-Untersegment, um die Ausnahmedetails anzuzeigen.



Die Ursache gibt an, dass die E-Mail-Adresse, die in einem Aufruf von `createSubscription` in der Klasse `WebConfig` angegeben wurde, ungültig ist. Im nächsten Abschnitt werden wir das beheben.

Konfigurieren von Amazon-SNS-Benachrichtigungen

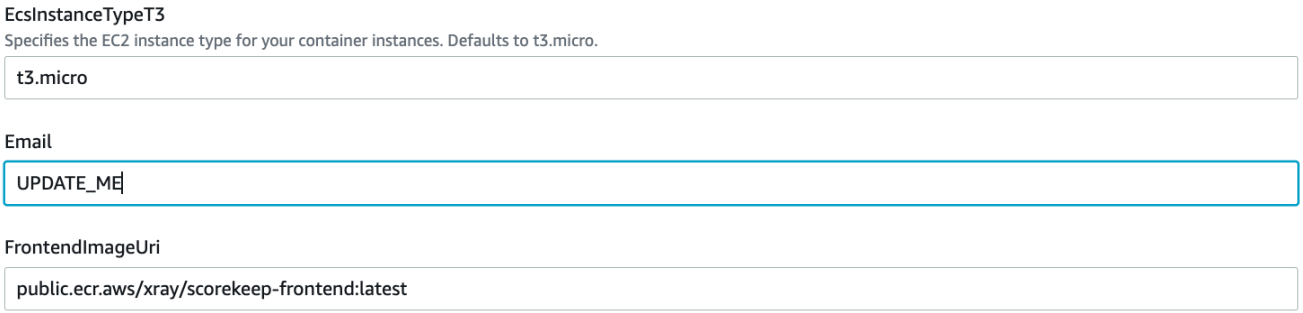
Scorekeep verwendet Amazon SNS, um Benachrichtigungen zu senden, wenn Benutzer ein Spiel abgeschlossen haben. Wenn die Anwendung gestartet wird, versucht sie, ein Abonnement für eine E-Mail-Adresse zu erstellen, die in einem CloudFormation Stack-Parameter definiert ist. Dieser Anruf schlägt derzeit fehl. Konfigurieren Sie eine Benachrichtigungs-E-Mail, um Benachrichtigungen zu aktivieren und die in der Trace-Map hervorgehobenen Fehler zu beheben.

AWS-Managementkonsole

Um Amazon SNS SNS-Benachrichtigungen mit dem zu konfigurieren AWS-Managementkonsole

1. Öffnen Sie die CloudFormation-Konsole.
2. Wählen Sie das Optionsfeld neben dem `scorekeep` Stack-Namen in der Liste und wählen Sie dann Aktualisieren.
3. Vergewissern Sie sich, dass Aktuelle Vorlage verwenden ausgewählt ist, und klicken Sie dann auf der Seite Stack aktualisieren auf Weiter.

4. Suchen Sie den E-Mail-Parameter in der Liste und ersetzen Sie den Standardwert durch eine gültige E-Mail-Adresse.



5. Scrollen Sie ans Seitenende und wählen Sie Next (Weiter).
6. Scrollen Sie zum Ende der Überprüfungsseite, aktivieren Sie das Kontrollkästchen zur Bestätigung, dass CloudFormation möglicherweise IAM-Ressourcen mit benutzerdefinierten Namen erstellt werden, und wählen Sie Stack aktualisieren aus.
7. Der CloudFormation Stack wird jetzt aktualisiert. Der Stack-Status wird `UPDATE_IN_PROGRESS` etwa fünf Minuten lang angezeigt, bevor er zu wechselt`UPDATE_COMPLETE`. Der Status wird regelmäßig aktualisiert, oder Sie können die Seite aktualisieren.

AWS CLI

Um Amazon SNS SNS-Benachrichtigungen mit dem zu konfigurieren AWS CLI

1. Navigieren Sie zu dem `xray-scorekeep/cloudformation/` Ordner, den Sie zuvor erstellt haben, und öffnen Sie die `cf-resources.yaml` Datei in einem Texteditor.
2. Suchen Sie den `Default` Wert im E-Mail-Parameter und ändern Sie ihn von *`UPDATE_ME`* in eine gültige E-Mail-Adresse.

```
Parameters:
  Email:
    Type: String
    Default: UPDATE_ME # <- change to a valid abc@def.xyz email address
```

3. Aktualisieren `cloudformation` Sie den CloudFormation Stack im Ordner mit dem folgenden AWS CLI Befehl:

```
aws cloudformation update-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

4. Warten SieUPDATE_COMPLETE, bis der CloudFormation Stack-Status lautet. Dies kann einige Minuten dauern. Verwenden Sie den folgenden AWS CLI Befehl, um den Status zu überprüfen:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Wenn die Aktualisierung abgeschlossen ist, startet Scorekeep neu und erstellt ein Abonnement für das SNS-Thema. Überprüfen Sie Ihre E-Mail-Adresse und bestätigen Sie das Abonnement, um Aktualisierungen anzuzeigen, wenn Sie ein Spiel abgeschlossen haben. Öffnen Sie die Trace-Map, um zu überprüfen, ob die Aufrufe an SNS nicht mehr fehlschlagen.

Erkunden der Beispielanwendung

Die Beispielanwendung ist eine HTTP-Web-API in Java, die für die Verwendung des X-Ray-SDK SDK for Java konfiguriert ist. Wenn Sie die Anwendung mit der CloudFormation Vorlage bereitstellen, erstellt sie die DynamoDB-Tabellen, den Amazon ECS-Cluster und andere Dienste, die für die Ausführung von Scorekeep auf ECS erforderlich sind. Eine Aufgabendefinitionsdatei für ECS wird über erstellt. CloudFormation Diese Datei definiert die Container-Images, die pro Aufgabe in einem ECS-Cluster verwendet werden. Diese Bilder stammen aus dem offiziellen öffentlichen X-Ray ECR. Das Scorekeep-API-Container-Image enthält die API, die mit Gradle kompiliert wurde. Das Container-Image des Scorekeep-Frontend-Containers bedient das Frontend mithilfe des Nginx-Proxyservers. Dieser Server leitet Anfragen an Pfade, die mit /api beginnen, an die API weiter.

Zum Instrumentieren eingehender HTTP-Anforderungen fügt die Anwendung den vom SDK bereitgestellten `TracingFilter` hinzu.

Example src/main/java/scorekeep/WebConfig.java — Servlet-Filter

```
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
...

@Configuration
public class WebConfig {
```

```
  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
...
```

Dieser Filter sendet Ablaufverfolgungsdaten über alle eingehenden Anforderungen, die die Anwendung verarbeitet, einschließlich Anforderungs-URL, Methode, Antwortstatus sowie Start- und Endzeit.

Die Anwendung führt auch Downstream-Aufrufe an DynamoDB mithilfe von durch. AWS SDK für Java Um diese Aufrufe zu instrumentieren, verwendet die Anwendung einfach die AWS SDK-bezogenen Submodule als Abhängigkeiten, und das X-Ray SDK for Java instrumentiert automatisch alle AWS SDK-Clients.

Die Anwendung verwendet`Docker`, um den Quellcode auf der Instanz mit der `Gradle Docker Image` und der `Scorekeep API Dockerfile` Datei zu erstellen, um die ausführbare JAR auszuführen, die Gradle in ihrer eigenen Datei generiert. `ENTRYPOINT`

Example Verwendung von Docker zum Erstellen über Gradle Docker Image

```
docker run --rm -v /PATH/TO/SCOREKEEP_REPO/home/gradle/project -w /home/gradle/project
 gradle:4.3 gradle build
```

Example Dockerfile ENTRYPOINT

```
ENTRYPOINT [ "sh", "-c", "java -Dserver.port=5000 -jar scorekeep-api-1.0.0.jar" ]
```

Die `build.gradle`-Datei lädt die SDK-Untermodule von Maven während der Kompilierung herunter, indem sie zu Abhängigkeiten erklärt werden.

Example build.gradle – Abhängigkeiten

```
...
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")
    }
}
```

Die Submodule Core, AWS SDK und AWS SDK Instrumentor sind alles, was erforderlich ist, um alle Downstream-Aufrufe, die mit dem SDK getätigt werden, automatisch zu instrumentieren. AWS

Um die Rohsegmentdaten an die X-Ray-API weiterzuleiten, muss der X-Ray-Daemon den Datenverkehr auf dem UDP-Port 2000 abhören. Zu diesem Zweck wird der X-Ray-Daemon in der Anwendung in einem Container ausgeführt, der zusammen mit der Scorekeep-Anwendung auf ECS als Sidecar-Container bereitgestellt wird. Weitere Informationen finden Sie im Thema X-Ray-Daemon.

Example X-Ray-Daemon-Container-Definition in einer ECS-Aufgabendefinition

```
...
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
      ...
```

Das X-Ray-SDK SDK for Java bietet eine Klasse mit dem Namen`AWSXRay`, die den Global Recorder bereitstellt`TracingHandler`, mit dem Sie Ihren Code instrumentieren können. Sie können die globale Aufzeichnung so konfigurieren, dass der `AWSXRayServletFilter`, der Segmente für

eingehende HTTP-Aufrufe erstellt, angepasst wird. Das Beispiel enthält einen statischen Block in der `WebConfig`-Klasse, der die globale Aufzeichnung mit Plugins und Samplingregeln konfiguriert.

Example src/main/java/scorekeep/WebConfig.java — Rekorder

```
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());
    ...

  }
}
```

In diesem Beispiel wird der Builder zum Laden von Samplingregeln aus einer Datei namens `sampling-rules.json` verwendet. Samplingregeln bestimmen die Rate, mit der das SDK Segmente für eingehende Anforderungen aufzeichnet.

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
  }
}
```

Die Samplingregeldatei definiert vier benutzerdefinierte Samplingregeln und die Standardregel. Für jede eingehende Anforderung wertet das SDK die benutzerdefinierten Regeln in der Reihenfolge aus, in der sie definiert sind. Das SDK wendet die erste Regel an, die der Methode, dem Pfad und dem Service-Namen der Anforderung entspricht. Bei Scorekeep fängt die erste Regel alle POST-Anforderungen (Aufrufe zum Erstellen von Ressourcen) ab, indem pro Sekunde ein festes Ziel einer einzelnen Anfrage und eine Rate von 1.0 angewendet wird bzw. 100 % der Anforderungen nach dem festen Ziel erfüllt sind.

Die übrigen drei benutzerdefinierte Regeln wenden eine Rate von fünf Prozent ohne festes Ziel auf Sitzungs-, Spiel- und Statuslesevorgänge (GET-Anfragen) an. Dies minimiert die Anzahl der Ablaufverfolgungen für regelmäßige Aufrufe, die das Front-End automatisch alle paar Sekunden ausführt, um sicherzustellen, dass die Inhalte auf dem neuesten Stand sind. Für alle übrigen Anforderungen definiert die Datei eine Standardrate von einer einzelnen Anfrage pro Sekunde und einer Rate von 10 Prozent.

Die Beispielanwendung zeigt auch, wie Sie erweiterte Funktionen wie die manuelle SDK-Client-Instrumentierung verwenden sowie zusätzliche Untersegmente und ausgehende HTTP-Aufrufe erstellen. Weitere Informationen finden Sie unter AWS X-Ray Musteranwendung.

Optional: Richtlinie für geringste Rechte

Die Scorekeep ECS-Container greifen mithilfe von Vollzugriffsrichtlinien wie und auf Ressourcen zu. `AmazonSNSFullAccess` `AmazonDynamoDBFullAccess` Die Verwendung von Richtlinien für vollen Zugriff ist nicht die beste Methode für Produktionsanwendungen. Im folgenden Beispiel wird die DynamoDB-IAM-Richtlinie aktualisiert, um die Sicherheit der Anwendung zu verbessern. Weitere Informationen zu bewährten Sicherheitsmethoden in IAM-Richtlinien finden Sie unter Identitäts- und Zugriffsmanagement für AWS X-Ray.

Example Vorlage cf-resources.yaml Rollendefinition 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"
```

Um Ihre Richtlinie zu aktualisieren, identifizieren Sie zunächst den ARN Ihrer DynamoDB-Ressourcen. Anschließend verwenden Sie den ARN in einer benutzerdefinierten IAM-Richtlinie. Schließlich wenden Sie diese Richtlinie auf Ihr Instanzprofil an.

So identifizieren Sie den ARN Ihrer DynamoDB-Ressource:

1. Öffnen Sie die DynamoDB-Konsole.
2. Wählen Sie in der linken Navigationsleiste Tabellen aus.
3. Wählen Sie eine der Optionen`scorekeep-*`, um die Tabellendetailseite anzuzeigen.
4. Wählen Sie auf der Registerkarte Übersicht die Option Zusätzliche Informationen aus, um den Abschnitt zu erweitern und den Amazon-Ressourcennamen (ARN) anzuzeigen. Kopieren Sie diesen Wert.
5. Fügen Sie den ARN in die folgende IAM-Richtlinie ein und ersetzen Sie die `AWS_ACCOUNT_ID` Werte `AWS_REGION` und durch Ihre spezifische Region und Konto-ID. Diese neue Richtlinie erlaubt nur die angegebenen Aktionen und nicht die `AmazonDynamoDBFullAccess` Richtlinie, die alle Aktionen zulässt.

   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-*"
        }
    ]
```

```
}
```

Die Tabellen, die über die Anwendung erstellt werden, folgen einer einheitlichen Namenskonvention. Sie können das `scorekeep-*` Format verwenden, um alle Scorekeep-Tabellen anzugeben.

Ändern Sie Ihre IAM-Richtlinie

1. Öffnen Sie die Scorekeep-Aufgabenrolle (ScorekeepRole) von der IAM-Konsole aus.
2. Aktivieren Sie das Kontrollkästchen neben der Richtlinie und wählen Sie Entfernen, um diese **`AmazonDynamoDBFullAccess`** Richtlinie zu entfernen.
3. Wählen Sie „Berechtigungen hinzufügen“, dann „Richtlinien anhängen“ und schließlich „Richtlinie erstellen“.
4. Wählen Sie die Registerkarte JSON und fügen Sie die oben erstellte Richtlinie ein.
5. Wählen Sie unten auf der Seite „Weiter: Tags“.
6. Wählen Sie unten auf der Seite „Weiter: Bewertung“.
7. Geben Sie unter Name einen Namen für die Richtlinie ein.
8. Wählen Sie unten auf der Seite die Option Richtlinie erstellen aus.
9. Hängen Sie die neu erstellte Richtlinie an die `scorekeepRole` Rolle an. Es kann einige Minuten dauern, bis die angehängte Richtlinie wirksam wird.

Wenn Sie die neue Richtlinie an die `scorekeepRole` Rolle angehängt haben, müssen Sie sie trennen, bevor Sie den CloudFormation Stapel löschen, da diese angehängte Richtlinie das Löschen des Stacks verhindert. Die Richtlinie kann automatisch getrennt werden, indem die Richtlinie gelöscht wird.

Entfernen Sie Ihre benutzerdefinierte IAM-Richtlinie

1. Öffnen Sie die IAM-Konsole.
2. Wählen Sie in der linken Navigationsleiste Richtlinien aus.
3. Suchen Sie nach dem benutzerdefinierten Richtliniennamen, den Sie zuvor in diesem Abschnitt erstellt haben, und wählen Sie das Optionsfeld neben dem Richtliniennamen, um ihn hervorzuheben.
4. Wählen Sie das Drop-down-Menü Aktionen und dann Löschen aus.

5. Geben Sie den Namen der benutzerdefinierten Richtlinie ein und wählen Sie dann Löschen, um den Löschvorgang zu bestätigen. Dadurch wird die Richtlinie automatisch von der `scorekeepRole` Rolle getrennt.

Bereinigen

Gehen Sie wie folgt vor, um die Ressourcen der Scorekeep-Anwendung zu löschen:

> **Note**
>
> Wenn Sie mithilfe des vorherigen Abschnitts dieses Tutorials benutzerdefinierte Richtlinien erstellt und angehängt haben, müssen Sie die Richtlinie aus dem entfernen, `scorekeepRole` bevor Sie den CloudFormation Stack löschen.

AWS-Managementkonsole

Löschen Sie die Beispielanwendung mithilfe des AWS-Managementkonsole

1. Öffnen Sie die CloudFormation-Konsole.
2. Wählen Sie das Optionsfeld neben dem `scorekeep` Stack-Namen in der Liste und wählen Sie dann Löschen aus.
3. Der CloudFormation Stapel wird jetzt gelöscht. Der Stack-Status bleibt `DELETE_IN_PROGRESS` einige Minuten lang bestehen, bis alle Ressourcen gelöscht sind. Der Status wird regelmäßig aktualisiert, oder Sie können die Seite aktualisieren.

AWS CLI

Löschen Sie die Beispielanwendung mit dem AWS CLI

1. Geben Sie den folgenden AWS CLI Befehl ein, um den CloudFormation Stack zu löschen:

   ```
   aws cloudformation delete-stack --stack-name scorekeep
   ```

2. Warten Sie, bis der CloudFormation Stapel nicht mehr existiert. Dies dauert etwa fünf Minuten. Verwenden Sie den folgenden AWS CLI Befehl, um den Status zu überprüfen:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Nächste Schritte

Erfahre mehr über X-Ray im nächsten Kapitel,AWS X-Ray Konzepte.

Um Ihre eigene App zu instrumentieren, erfahren Sie mehr über das X-Ray SDK for Java oder eines der anderen X-Ray SDKs:

- X-Ray SDK for Java — AWS X-Ray SDK for Java
- X-Ray SDK für Node.js — AWS X-Ray-SDK für Node.js
- X-Ray SDK for .NET — AWS X-Ray SDK for .NET

Informationen zum lokalen oder eingeschalteten Ausführen des X-Ray-Daemons finden Sie unterAWS X-Ray Dämon. AWS

Wie Sie zur Beispielanwendung beitragen können GitHub, finden Sie unter eb-java-scorekeep.

Manuelles Instrumentieren von AWS SDK-Clients

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK SDK for Java instrumentiert automatisch alle AWS SDK-Clients, wenn Sie das AWS SDK Instrumentor-Untermodul in Ihre Build-Abhängigkeiten aufnehmen.

Sie können die automatische Client-Instrumentierung durch Löschen des Instrumentor-Untermoduls deaktivieren. Auf diese Weise können Sie einige Clients manuell instrumentieren und andere ignorieren oder verschiedene Tracing-Handler auf unterschiedlichen Clients anwenden.

Um die Unterstützung für die Instrumentierung bestimmter AWS SDK-Clients zu veranschaulichen, übergibt die Anwendung einen Tracing-Handler `AmazonDynamoDBClientBuilder` als Anforderungshandler im Benutzer-, Spiel- und Sitzungsmodell. Diese Codeänderung weist das SDK an, alle Aufrufe von DynamoDB mithilfe dieser Clients zu instrumentieren.

Example **`src/main/java/scorekeep/SessionModel.java`**— Manuelle AWS SDK-Client-Instrumentierung

```
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);
```

Wenn Sie das AWS SDK Instrumentor-Untermodul aus den Projektabhängigkeiten entfernen, werden nur die manuell instrumentierten AWS SDK-Clients in der Trace-Map angezeigt.

Erstellen zusätzlicher Untersegmente

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

In der Benutzermodellklasse erstellt die Anwendung manuell Untersegmente, um alle nachgelagerten Aufrufe, die innerhalb der `saveUser`-Funktion vorgenommen wurden, zu gruppieren, und fügt Metadaten hinzu.

Example **src/main/java/scorekeep/UserModel.java** – Benutzerdefinierte Untersegmente

```
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();
    }
  }
```

Anmerkungen, Metadaten und Benutzer aufzeichnen IDs

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

In der Spielmodellklasse zeichnet die Anwendung jedes Mal, wenn sie ein Spiel in DynamoDB speichert, `Game` Objekte in einem Metadatenblock auf. Unabhängig davon zeichnet die Anwendung das Spiel IDs in Anmerkungen zur Verwendung mit Filterausdrücken auf.

Example **src/main/java/scorekeep/GameModel.java**— Anmerkungen und Metadaten

```
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();
    }
  }
```

Im Move-Controller zeichnet die Anwendung den Benutzer IDs mit `setUser` auf. Benutzer IDs werden in einem separaten Feld in Segmenten aufgezeichnet und für die Verwendung bei der Suche indexiert.

Example src/main/java/scorekeep/MoveController.java — Benutzer-ID

```
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);
```

```
  }
```

Instrumentieren von ausgehenden HTTP-Aufrufen

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Die User-Factory-Klasse zeigt, wie die Anwendung das X-Ray-SDK für die Java-Version von verwendetHTTPClientBuilder, um ausgehende HTTP-Aufrufe zu instrumentieren.

Example **src/main/java/scorekeep/UserFactory.java**— HTTPClient Instrumentierung

```
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();
    }
  }
```

Wenn Sie derzeit `org.apache.http.impl.client.HttpClientBuilder` verwenden, können Sie einfach die Import-Anweisung für die Klasse mit einer Anweisung für `com.amazonaws.xray.proxies.apache.http.HttpClientBuilder` austauschen.

Instrumentieren von Aufrufen einer PostgreSQL-Datenbank

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Die `application-pgsql.properties` Datei fügt der in erstellten Datenquelle den X-Ray PostgreSQL-Tracing-Interceptor hinzu. `RdsWebConfig.java`

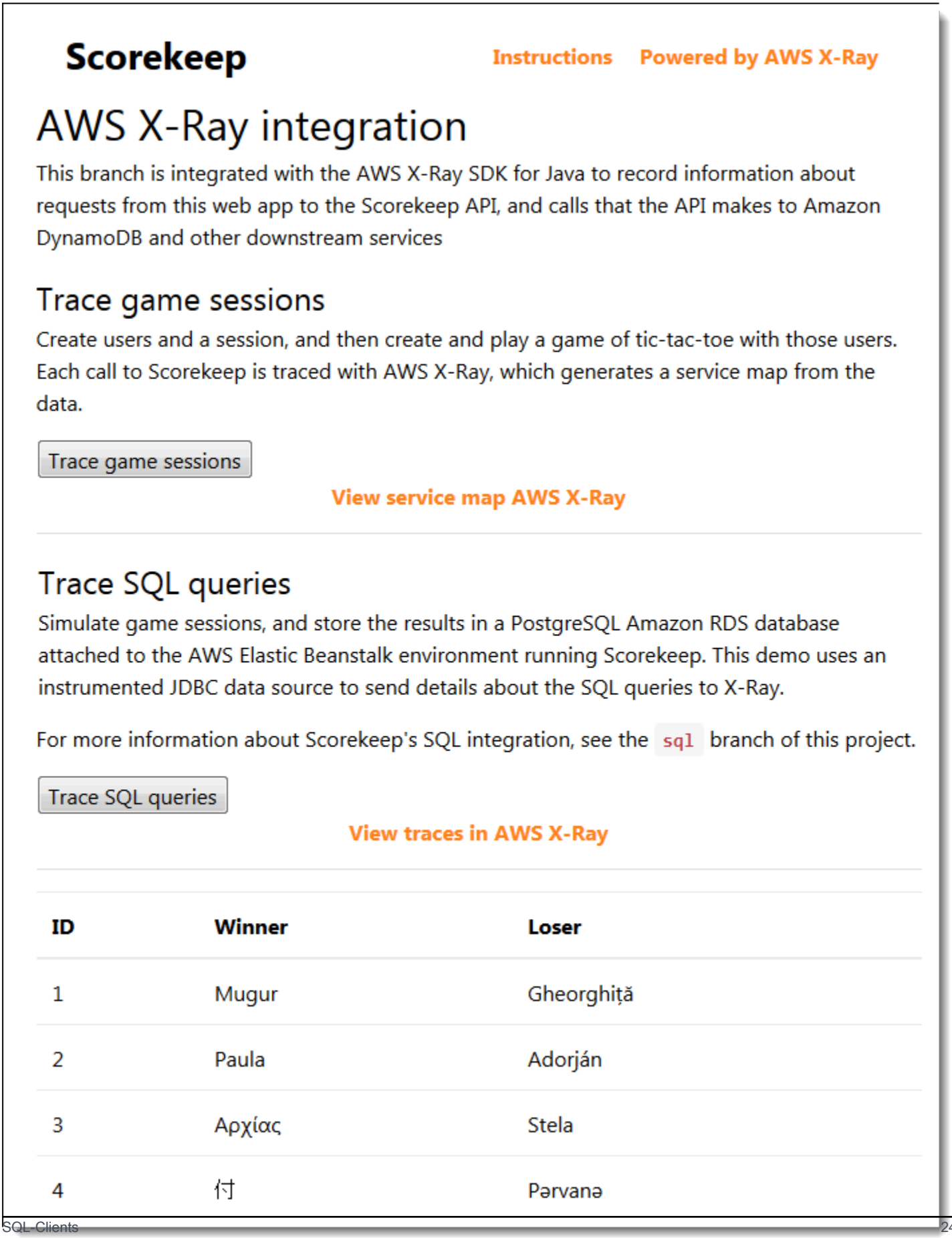
Example **`application-pgsql.properties`**— PostgreSQL-Datenbankinstrumentierung

```
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**
>
> Weitere Informationen zum Hinzufügen einer PostgreSQL-Datenbank zum Anwendungsumfeld finden Sie unter Konfiguration von Datenbanken mit Elastic Beanstalk im AWS Elastic Beanstalk -Entwicklerhandbuch.

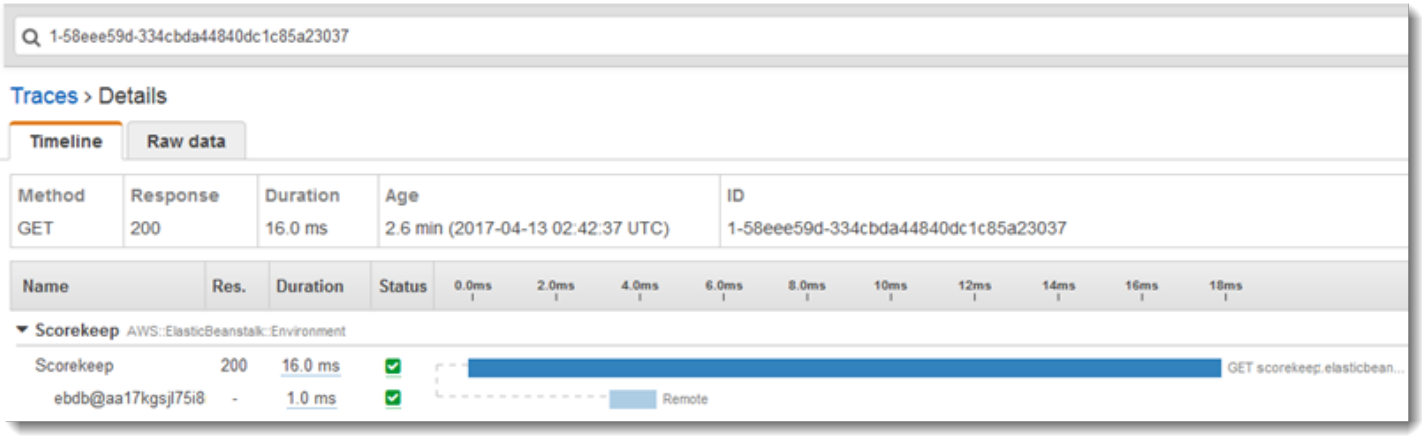
Die X-Ray-Demoseite in der `xray` Branche enthält eine Demo, die die instrumentierte Datenquelle verwendet, um Traces zu generieren, die Informationen über die von ihr generierten SQL-Abfragen

enthalten. Navigieren Sie zu dem `/#/xray`-Pfad in der laufenden Anwendung oder wählen Sie in der Navigationsleiste Powered by AWS X-Ray aus, um die Demo-Seite anzusehen.


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

Wählen Sie SQL-Abfragen nachverfolgen aus, um Spielsitzungen zu simulieren und die Ergebnisse in der zugehörigen Datenbank zu speichern. Wählen Sie dann „Traces in AWS X-Ray anzeigen“, um eine gefilterte Liste der Traces anzuzeigen, die auf die `/api/history` Route der API zutreffen.

Wählen Sie eine der Ablaufverfolgungen aus der Liste aus, um die Zeitleiste, einschließlich der SQL-Abfrage, ansehen zu können.



Instrumentierungsfunktionen AWS Lambda

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Scorekeep verwendet zwei Funktionen. AWS Lambda Bei der ersten handelt es sich um eine Node.js-Funktion der `lambda`-Verzweigung, die zufällige Namen für neue Benutzer generiert. Wenn ein Benutzer eine Sitzung erstellt, ohne einen Namen einzugeben, ruft die Anwendung eine Funktion namens `random-name` mit dem AWS SDK für Java auf. Das X-Ray SDK for Java zeichnet Informationen über den Aufruf von Lambda in einem Untersegment auf, wie jeder andere Aufruf, der mit einem instrumentierten AWS SDK-Client getätigt wird.

> **Note**
>
> Die Ausführung der `random-name` Lambda-Funktion erfordert die Erstellung zusätzlicher Ressourcen außerhalb der Elastic Beanstalk Beanstalk-Umgebung. Weitere Informationen und Anweisungen finden Sie in der Readme-Datei: AWS Lambda Integration.

Die zweite Funktion, `scorekeep-worker`, ist eine Python-Funktion, die unabhängig von der Scorekeep-API ausgeführt wird. Wenn ein Spiel endet, schreibt die API die Sitzungs- und Spiele-ID in eine SQS-Warteschlange. Die Worker-Funktion liest Elemente aus der Warteschlange und ruft die Scorekeep-API auf, um vollständige Aufzeichnungen jeder Spielsitzung für die Speicherung in Amazon S3 zu erstellen.

Scorekeep enthält CloudFormation Vorlagen und Skripte zur Erstellung beider Funktionen. Da Sie das X-Ray-SDK mit dem Funktionscode bündeln müssen, erstellen die Vorlagen die Funktionen ohne Code. Wenn Sie Scorekeep bereitstellen, erstellt eine im Ordner `.ebextensions` enthaltene Konfigurationsdatei ein Quell-Bundle, in dem das SDK enthalten ist, und aktualisiert den Funktionscode und die Konfiguration mit der AWS Command Line Interface.

Funktionen

- Zufälliger Name
- Worker

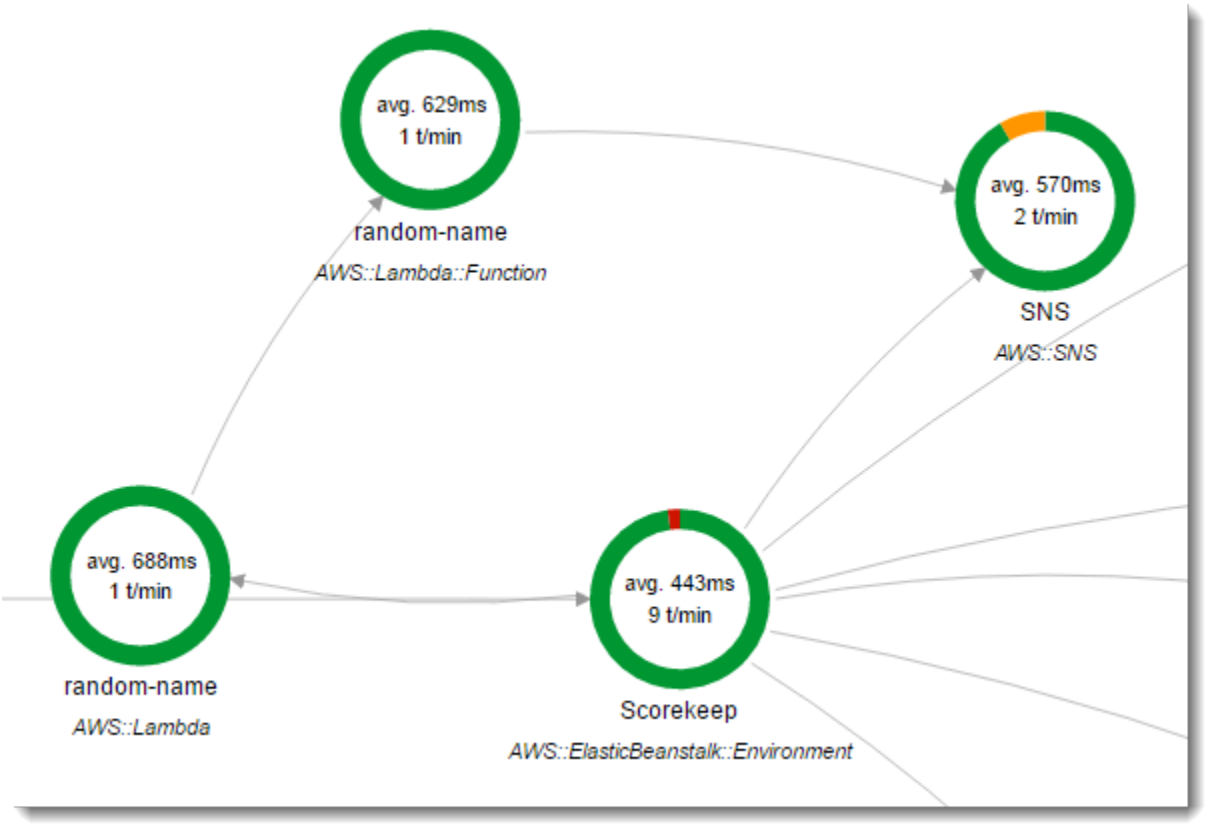
Zufälliger Name

Scorekeep ruft die Funktion für zufällige Namen auf, wenn ein Benutzer eine Spielesitzung beginnt, ohne dass er sich anmeldet oder einen Benutzernamen angibt. Wenn Lambda den Aufruf von verarbeitet`random-name`, liest es den Tracing-Header, der die vom X-Ray SDK for Java geschriebene Trace-ID und die Sampling-Entscheidung enthält.

Für jede gesampelte Anfrage führt Lambda den X-Ray-Daemon aus und schreibt zwei Segmente. Das erste Segment zeichnet Informationen über den Aufruf von Lambda auf, der die Funktion aufruft. Dieses Segment enthält dieselben Informationen wie das von Scorekeep aufgezeichnete Untersegment, jedoch aus Lambda-Sicht. Das zweite Segment repräsentiert die von der Funktion durchgeführte Arbeit.

Lambda übergibt das Funktionssegment über den Funktionskontext an das X-Ray SDK. Wenn Sie eine Lambda-Funktion instrumentieren, verwenden Sie das SDK nicht, um ein Segment für

eingehende Anfragen zu erstellen. Lambda stellt das Segment bereit, und Sie verwenden das SDK, um Clients zu instrumentieren und Untersegmente zu schreiben.



Die `random-name`-Funktion wird in Node.js implementiert. Es verwendet das SDK für JavaScript in Node.js, um Benachrichtigungen mit Amazon SNS zu senden, und das X-Ray SDK für Node.js, um den SDK-Client zu AWS instrumentieren. Zum Schreiben von Anmerkungen erstellt die Funktion ein benutzerdefiniertes Untersegment mit `AWSXRay.captureFunc` und schreibt die Anmerkungen in die instrumentierte Funktion. In Lambda können Sie Anmerkungen nicht direkt in das Funktionssegment schreiben, sondern nur in ein Untersegment, das Sie erstellen.

Example **`function/index.js`** – Zufallsname-Lambda-Funktion

```
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;
```

Diese Funktion wird automatisch erstellt, wenn Sie die Beispielanwendung für Elastic Beanstalk bereitstellen. Der `xray` Zweig enthält ein Skript zum Erstellen einer leeren Lambda-Funktion. Die Konfigurationsdateien im `.ebextensions` Ordner erstellen das Funktionspaket mit `npm install` während der Bereitstellung und aktualisieren dann die Lambda-Funktion mit der AWS CLI.

Worker

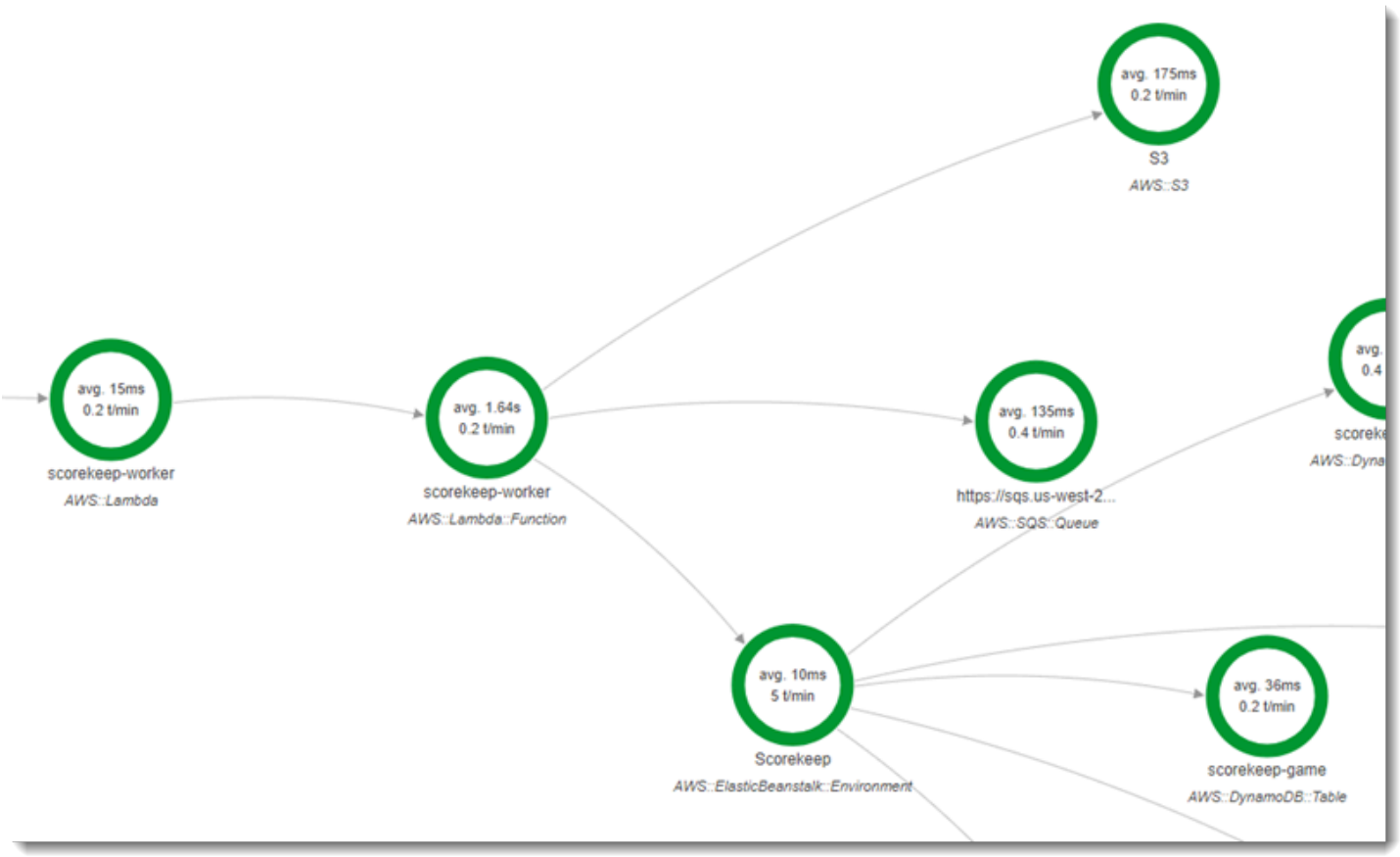
Die instrumentierten Worker-Funktion wird in einer eigenen Verzweigung, `xray-worker`, bereitgestellt, da sie nur ausgeführt werden kann, wenn Sie die Worker-Funktion und verwandte Ressourcen zuerst erstellen. Detaillierte Anweisungen finden Sie in der Readme-Datei zur Verzweigung.

Die Funktion wird alle 5 Minuten durch ein gebündeltes Amazon CloudWatch Events-Ereignis ausgelöst. Wenn sie ausgeführt wird, ruft die Funktion ein Element aus einer Amazon SQS SQS-

Warteschlange ab, die von Scorekeep verwaltet wird. Jede Nachricht enthält Informationen zu einem abgeschlossenen Spiel.

Der Worker ruft den Spieledatensatz und die Spieledokumente von anderen Tabellen ab, auf die der Spieledatensatz verweist. Beispielsweise enthält der Spieldatensatz in DynamoDB eine Liste der Züge, die während des Spiels ausgeführt wurden. Die Liste enthält nicht die Züge selbst, sondern IDs Züge, die in einer separaten Tabelle gespeichert sind.

Sitzungen und Status werden ebenfalls als Referenzen gespeichert. Auf diese Weise wird verhindert, dass die Einträge in der Spieletabelle zu groß werden. Allerdings sind zusätzliche Aufrufe notwendig, um alle Informationen zum Spiel zu erhalten. Der Worker dereferenziert all diese Einträge und erstellt eine vollständige Aufzeichnung des Spiels als einzelnes Dokument in Amazon S3. Wenn Sie die Daten analysieren möchten, können Sie Abfragen direkt in Amazon S3 mit Amazon Athena ausführen, ohne leseintensive Datenmigrationen durchführen zu müssen, um Ihre Daten aus DynamoDB zu holen.



Für die Worker-Funktion ist die aktive Nachverfolgung in der Konfiguration in AWS Lambda aktiviert. Im Gegensatz zur Funktion mit zufälligen Namen erhält der Worker keine Anfrage von einer

instrumentierten Anwendung und somit auch AWS Lambda keinen Tracing-Header. Bei aktivem Tracing erstellt Lambda die Trace-ID und trifft Stichprobenentscheidungen.

Das X-Ray-SDK für Python ist nur ein paar Zeilen am Anfang der Funktion, die das SDK importieren und seine `patch_all` Funktion zum Patchen ausführen AWS SDK für Python (Boto) und HTTclients die zum Aufrufen von Amazon SQS und Amazon S3 verwendet werden. Wenn der Auftragnehmer die Scorekeep-API aufruft, fügt das SDK den Ablaufverfolgungs-Header zur Anforderung hinzu, um Aufrufe über die API nachzuverfolgen.

Example_lambda/scorekeep-worker/scorekeep-worker.py – Worker Lambda-Funktion

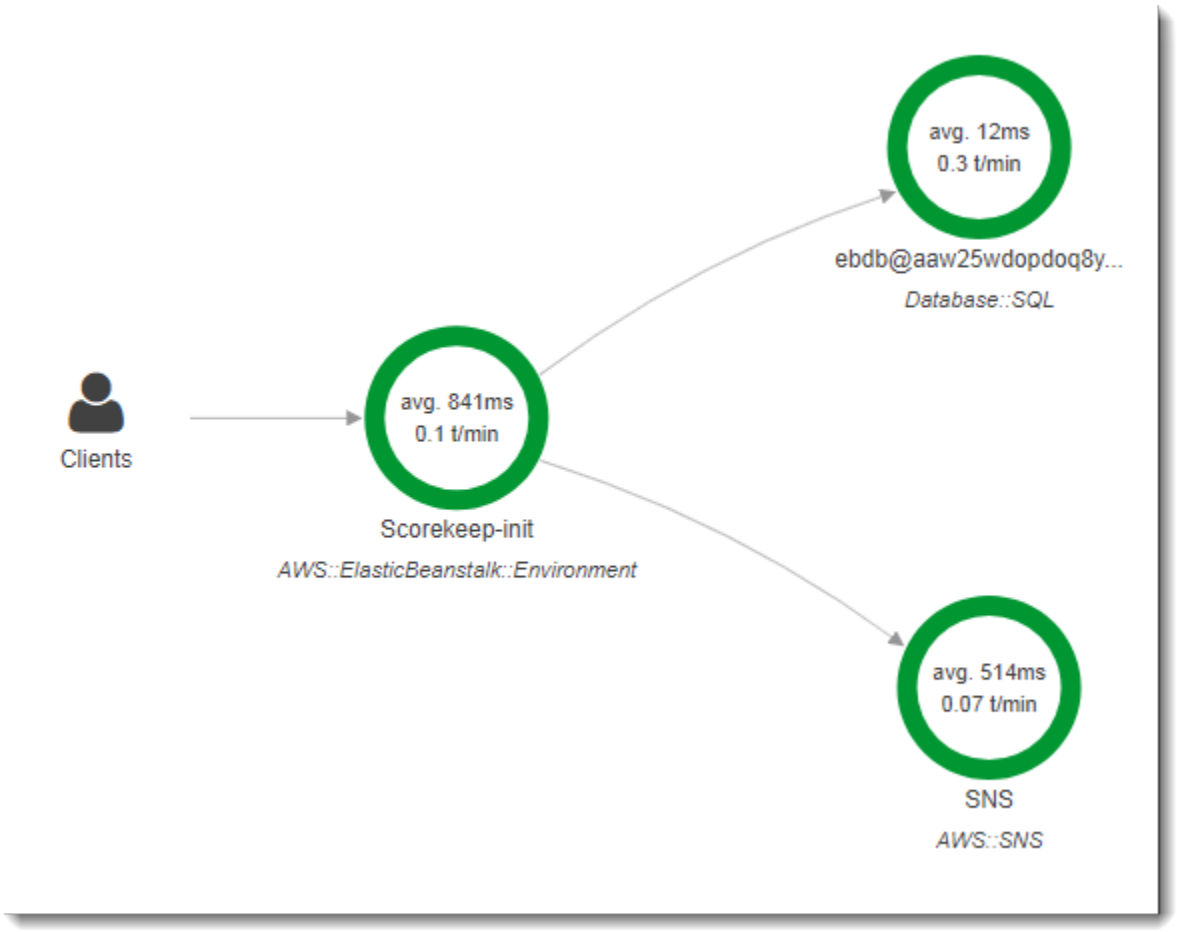
```
import os
import boto3
import json
import requests
import time
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

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
    )
   ...
```

Instrumentieren von Startup-Code

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray SDK for Java erstellt automatisch Segmente für eingehende Anfragen. Wenn eine Anfrage im Leistungsumfang enthalten ist, können Sie instrumentierte Clients verwenden und Untersegmente ohne Probleme aufzeichnen. Wenn Sie jedoch versuchen, einen instrumentierten Client im Startcode zu verwenden, erhalten Sie einen SegmentNotFoundException.

Startcode wird außerhalb des request/response Standardablaufs einer Webanwendung ausgeführt, sodass Sie Segmente manuell erstellen müssen, um ihn zu instrumentieren. Scorekeep stellt die Instrumentierung von Startup-Code in den `WebConfig`-Dateien dar. Scorekeep ruft beim Start eine SQL-Datenbank und Amazon SNS auf.



Die `WebConfig` Standardklasse erstellt ein Amazon SNS SNS-Abonnement für Benachrichtigungen. Um ein Segment bereitzustellen, in das das X-Ray-SDK schreiben kann, wenn der Amazon SNS-Client verwendet wird, ruft Scorekeep `endSegment` auf `beginSegment` und auf dem Global Recorder auf.

Example **`src/main/java/scorekeep/WebConfig.java`**— Instrumentierter AWS SDK-Client im Startcode

```
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();
```

In`RdsWebConfig`, das Scorekeep verwendet, wenn eine Amazon RDS-Datenbank verbunden ist, erstellt die Konfiguration auch ein Segment für den SQL-Client, den Hibernate verwendet, wenn es das Datenbankschema beim Start anwendet.

Example **`src/main/java/scorekeep/RdsWebConfig.java`**— Instrumentierter SQL-Datenbank-Client im Startcode

```
@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` wird automatisch ausgeführt und verwendet einen SQL-Client. Da der Client instrumentiert ist, muss Scorekeep die Standardimplementierung überschreiben und ein Segment bereitstellen, das das SDK verwenden kann, wenn der Client aufgerufen wird.

Instrumentieren von Skripten

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon

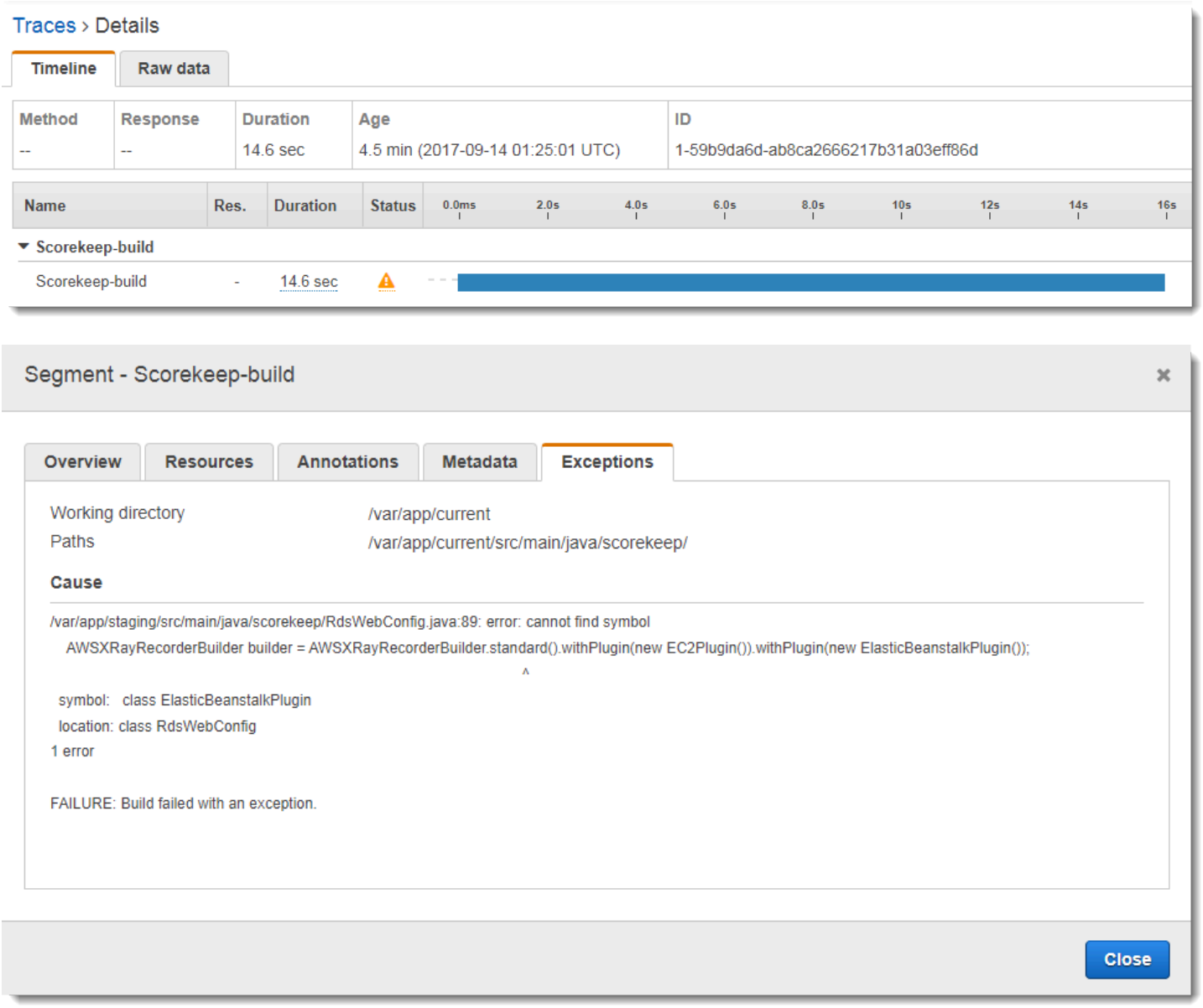
> Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können auch Code instrumentieren, der nicht Teil Ihrer Anwendung ist. Wenn der X-Ray-Daemon läuft, leitet er alle Segmente, die er empfängt, an X-Ray weiter, auch wenn sie nicht vom X-Ray-SDK generiert wurden. Scorekeep nutzt seine eigenen Skripte zur Instrumentierung des Builds, der die Anwendung während der Bereitstellung kompiliert.

Example **`bin/build.sh`**— Instrumentiertes Build-Skript

```
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` und `xray_success.py` sind einfache Python-Skripte, die Segmentobjekte erstellen, diese in JSON-Dokumente konvertieren und über UDP an den Daemon senden. Wenn der Gradle-Build fehlschlägt, können Sie die Fehlermeldung finden, indem Sie in der Trace-Map der X-Ray-Konsole auf den Scorekeep-Build-Knoten klicken.



Instrumentieren eines Web-App-Clients

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon
> Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu

OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

In der `xray-cognito`Filiale verwendet Scorekeep Amazon Cognito, damit Benutzer ein Konto erstellen und sich damit anmelden können, um ihre Benutzerinformationen aus einem Amazon Cognito Cognito-Benutzerpool abzurufen. Wenn sich ein Benutzer anmeldet, verwendet Scorekeep einen Amazon Cognito Cognito-Identitätspool, um temporäre AWS Anmeldeinformationen für die Verwendung mit dem zu erhalten. AWS SDK für JavaScript

Der Identitätenpool ist so konfiguriert, dass angemeldete Benutzer Ablaufverfolgungsdaten in AWS X-Ray schreiben können. Die Web-App nutzt diese Anmeldeinformationen, um die Benutzer-ID des angemeldeten Benutzers, den Browserpfad und die Client-Ansicht von Aufrufen der Scorekeep-API aufzuzeichnen.

Der Großteil der Vorgänge wird in einer Service-Klasse mit dem Namen `xray` ausgeführt. Diese Serviceklasse bietet Methoden zum Generieren der erforderlichen Identifikatoren, zum Erstellen von Segmenten in Bearbeitung, zum Finalisieren von Segmenten und zum Senden von Segmentdokumenten an die X-Ray-API.

Example **`public/xray.js`**— Segmente aufzeichnen und hochladen

```
...
  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);
    }
  })
}
```

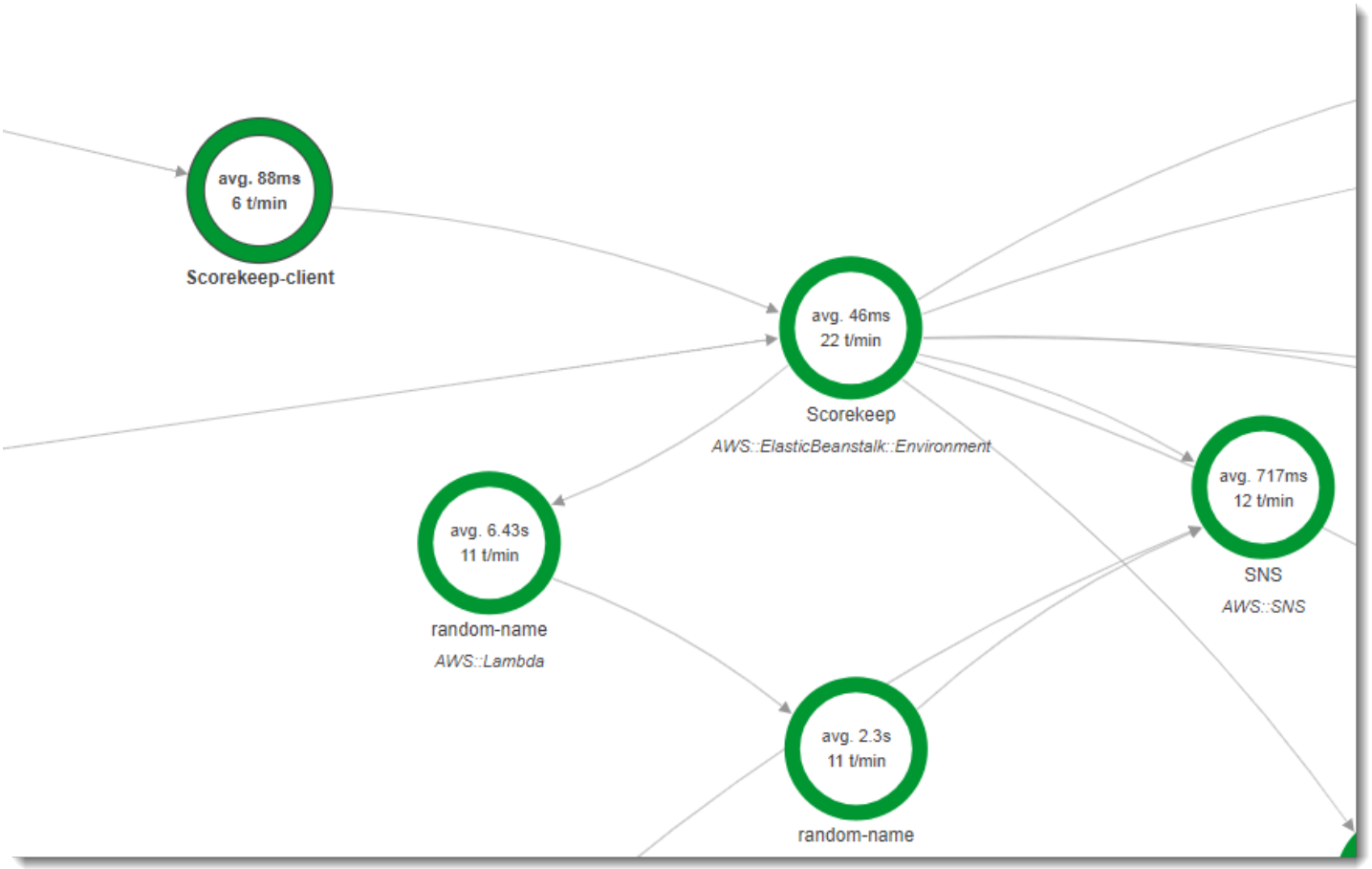
Diese Methoden werden im Header und in `transformResponse`-Funktionen in den Ressourcen-Services aufgerufen, die die Web-App zum Aufrufen der Scorekeep-API verwendet. Um das Clientsegment in denselben Trace aufzunehmen wie das von der API generierte Segment, muss die Web-App die Trace-ID und die Segment-ID in einen Tracing-Header (`X-Amzn-Trace-Id`) aufnehmen, den das X-Ray-SDK lesen kann. Wenn die instrumentierte Java-Anwendung eine Anfrage mit diesem Header empfängt, verwendet das X-Ray SDK for Java dieselbe Trace-ID und macht das Segment vom Web-App-Client zum übergeordneten Segment.

Example **`public/app/services.js`**— Aufzeichnen von Segmenten für Angular Resource Calls und Schreiben von Tracing-Headern

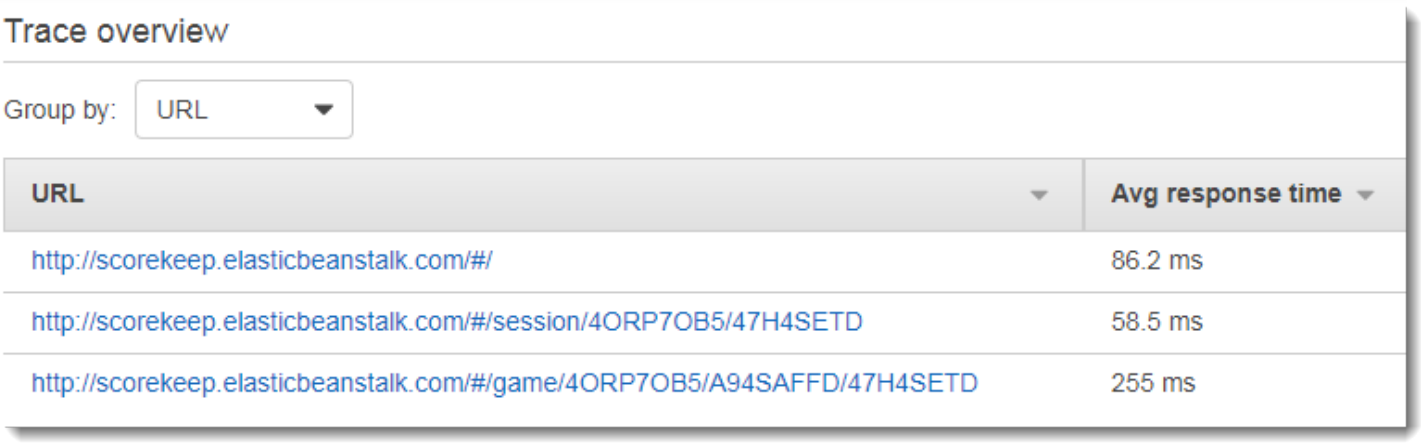
```
var module = angular.module('scorekeep');
```

```
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);
      },
    },
...
```

Die resultierende Trace-Map enthält einen Knoten für den Web-App-Client.



Ablaufverfolgungen, die Segmente aus der Web-App einschließen, zeigen die URL an, die der Benutzer im Browser sieht (Pfad beginnend mit `/#/`). Ohne Client-Instrumentierung erhalten Sie nur die URL der API-Ressource, die die Web-App aufruft (Pfade beginnend mit `/api/`).



Verwenden instrumentierter Clients in Auftragnehmer-Threads

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Scorekeep verwendet einen Worker-Thread, um eine Benachrichtigung an Amazon SNS zu veröffentlichen, wenn ein Benutzer ein Spiel gewinnt. Die Veröffentlichung der Benachrichtigung dauert länger als alle übrigen Anfragevorgänge zusammen und wirkt sich nicht auf den Client oder Benutzer aus. Die Aufgabe asynchron auszuführen ist daher eine gute Möglichkeit, die Reaktionszeit zu verbessern.

Das X-Ray-SDK SDK for Java weiß jedoch nicht, welches Segment aktiv war, als der Thread erstellt wurde. Wenn Sie also versuchen, den instrumentierten AWS SDK für Java Client innerhalb des Threads zu verwenden, wird ein `SegmentNotFoundException` Fehler ausgelöst, wodurch der Thread abstürzt.

Example Web-1.error.log

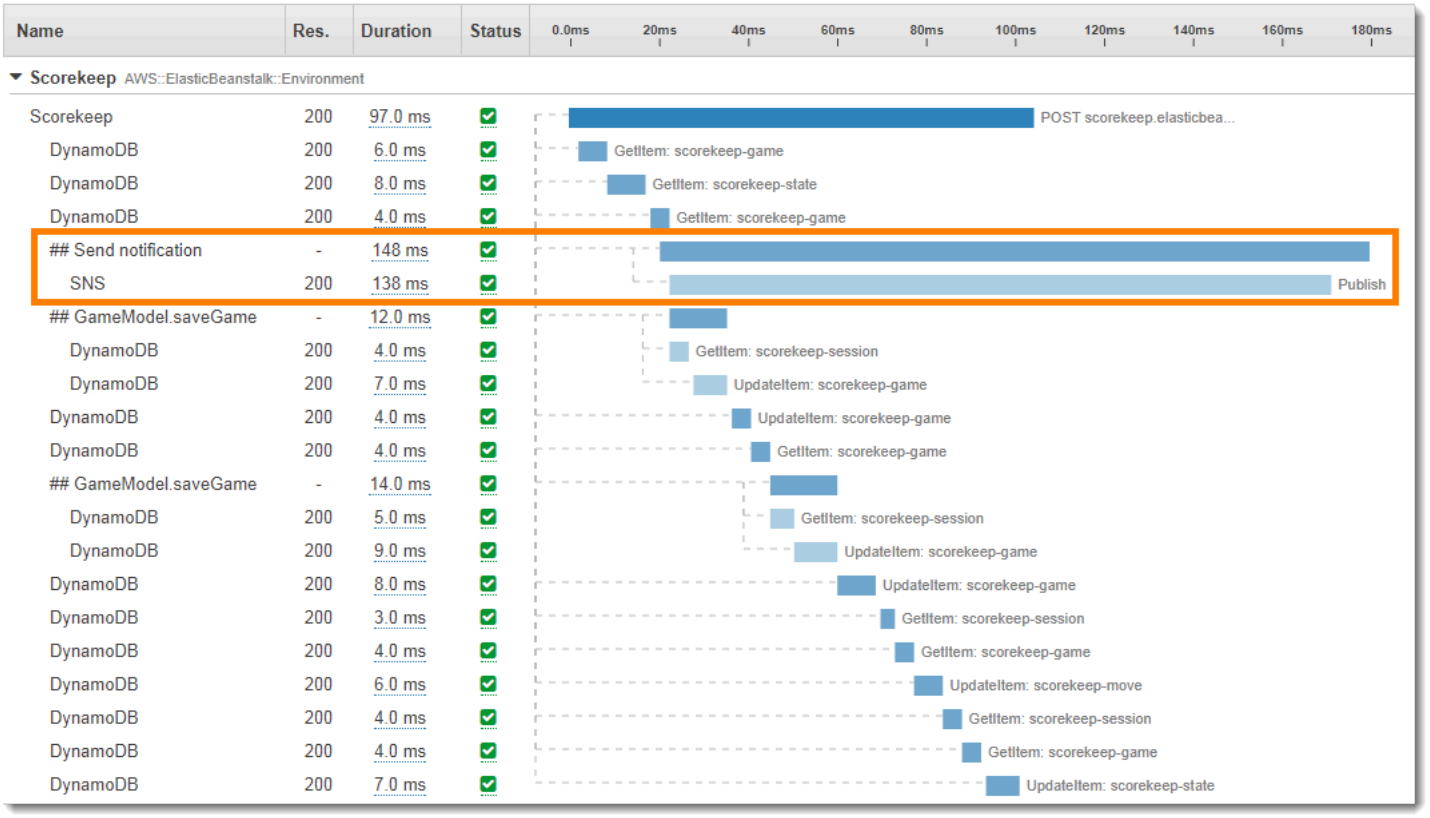
```
Exception in thread "Thread-2" com.amazonaws.xray.exceptions.SegmentNotFoundException:
 Failed to begin subsegment named 'AmazonSNS': segment cannot be found.
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at
 sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at
 sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
...
```

Um dieses Problem zu beheben, verwendet die Anwendung, `GetTraceEntity` um einen Verweis auf das Segment im Haupt-Thread abzurufen und `Entity.run()` den Worker-Thread-Code mit Zugriff auf den Kontext des Segments sicher auszuführen.

Example **src/main/java/scorekeep/MoveFactory.java**— Übergibt den Trace-Kontext an einen Worker-Thread

```
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();
          }
        }
```

Da die Anfrage nun vor dem Aufruf von Amazon SNS gelöst wird, erstellt die Anwendung ein separates Untersegment für den Thread. Dadurch wird verhindert, dass das X-Ray-SDK das Segment schließt, bevor es die Antwort von Amazon SNS aufzeichnet. Wenn zum Zeitpunkt der Bearbeitung der Anfrage durch Scorekeep kein Untersegment geöffnet war, ging die Antwort von Amazon SNS möglicherweise verloren.



Weitere Informationen zu Multithreading finden Sie unter Übermitteln von Segmentkontext zwischen Threads in einer Multithread-Anwendung.

AWS X-Ray Dämon

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

> **Note**
>
> Sie können den CloudWatch Agenten jetzt verwenden, um Metriken, Protokolle und Traces von EC2 Amazon-Instances und lokalen Servern zu sammeln. CloudWatch Agentenversion 1.300025.0 und höher können Traces von OpenTelemetryunserem X-Ray-Client SDKs sammeln und an X-Ray senden. Wenn Sie den CloudWatch Agenten anstelle des AWS Distro for OpenTelemetry (ADOT) Collector oder des X-Ray-Daemons zum Sammeln von Traces verwenden, können Sie die Anzahl der verwalteten Agenten reduzieren. Weitere Informationen finden Sie unter dem Thema CloudWatch Agent im CloudWatch Benutzerhandbuch.

Der AWS X-Ray Daemon ist eine Softwareanwendung, die den Datenverkehr auf dem UDP-Port 2000 überwacht, Rohsegmentdaten sammelt und sie an die API weiterleitet. AWS X-Ray Der Daemon arbeitet mit dem zusammen AWS X-Ray SDKs und muss laufen, damit die vom gesendeten Daten den X-Ray-Dienst erreichen SDKs können. Der X-Ray-Daemon ist ein Open-Source-Projekt. Du kannst dem Projekt folgen und Issues und Pull-Requests einreichen auf GitHub: github. com/aws/aws-Xray-Daemon

Aktivieren AWS Lambda und AWS Elastic Beanstalk verwenden Sie die Integration dieser Dienste mit X-Ray, um den Daemon auszuführen. Lambda führt den Daemon jedes Mal automatisch aus, wenn eine Funktion für eine Stichprobenanforderung aufgerufen wird. Verwenden Sie auf Elastic Beanstalk die `XRayEnabled` Konfigurationsoption, um den Daemon auf den Instances in Ihrer Umgebung auszuführen. Die Ausgabe des obigen Befehls sieht in etwa folgendermaßen aus (JSON format).

Um den X-Ray-Daemon lokal, lokal oder auf einem anderen Server auszuführen, laden Sie ihn herunter AWS-Services, führen Sie ihn aus und geben Sie ihm dann die Erlaubnis, Segmentdokumente auf X-Ray hochzuladen.

Herunterladen des Daemons

Sie können den Daemon von Amazon S3, Amazon ECR oder Docker Hub herunterladen und ihn dann lokal ausführen oder ihn beim Start auf einer EC2 Amazon-Instance installieren.

Amazon S3

X-Ray-Daemon-Installationsprogramme und ausführbare Dateien

- Linux (ausführbar) — (sig) **`aws-xray-daemon-linux-3.x.zip`**
- Linux (RPM-Installationsprogramm) — `aws-xray-daemon-3.x.rpm`
- Linux (DEB-Installationsprogramm) — `aws-xray-daemon-3.x.deb`
- Linux (ARM64, ausführbar) — `aws-xray-daemon-linux-arm64-3.x.zip`(sig)
- Linux (ARM64, RPM-Installationsprogramm) — `aws-xray-daemon-arm64-3.x.rpm`
- Linux (ARM64, DEB-Installationsprogramm) — `aws-xray-daemon-arm64-3.x.deb`
- OS X (ausführbar) — `aws-xray-daemon-macos-3.x.zip`(sig)
- Windows (ausführbar) — `aws-xray-daemon-windows-process-3.x.zip`(sig)
- Windows (Dienst) — `aws-xray-daemon-windows-service-3.x.zip`(sig)

Diese Links verweisen immer auf die neueste 3.x-Version des Daemons. Gehen Sie wie folgt vor, um eine bestimmte Version herunterzuladen:

- Wenn Sie eine Version vor der Version herunterladen möchten`3.3.0`, `3.x` ersetzen Sie sie durch die Versionsnummer. Beispiel, `2.1.0`. Vor der Version `3.3.0` ist die einzig verfügbare Architektur`arm64`. Beispiel: `2.1.0` und `arm64`.
- Wenn Sie eine Version nach der anderen herunterladen möchten`3.3.0`, `3.x` ersetzen Sie diese durch die Versionsnummer und `arch` den Architekturtyp.

X-Ray-Assets werden in jeder unterstützten Region in Buckets repliziert. Um den Bucket zu verwenden, der Ihnen oder Ihren AWS Ressourcen am nächsten ist, ersetzen Sie die Region in den obigen Links durch Ihre Region.

```
https://s3.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-daemon/aws-xray-daemon-3.x.rpm
```

Amazon ECR

Ab Version 3.2.0 ist der Daemon auf Amazon ECR zu finden. Bevor Sie ein Image abrufen, sollten Sie Ihren Docker-Client bei der öffentlichen Registrierung von Amazon ECR authentifizieren.

Rufen Sie das neueste veröffentlichte 3.x-Versions-Tag ab, indem Sie den folgenden Befehl ausführen:

```
docker pull public.ecr.aws/xray/aws-xray-daemon:3.x
```

Frühere Versionen oder Alpha-Versionen können heruntergeladen werden, indem sie durch eine `alpha` oder eine bestimmte Versionsnummer ersetzt `3.x` werden. Es wird nicht empfohlen, in einer Produktionsumgebung ein Daemon-Image mit einem Alpha-Tag zu verwenden.

Docker Hub

Der Daemon ist auf Docker Hub zu finden. Führen Sie den folgenden Befehl aus, um die neueste veröffentlichte 3.x-Version herunterzuladen:

```
docker pull amazon/aws-xray-daemon:3.x
```

Frühere Versionen des Daemons können veröffentlicht werden, indem sie durch die gewünschte `3.x` Version ersetzt werden.

Überprüfen der Signatur des Daemon-Archivs

GPG-Signatur-Dateien sind für Daemon-Komponenten in komprimierten ZIP-Archiven eingeschlossen. Der öffentliche Schlüssel befindet sich hier: `aws-xray.gpg`.

Sie können den öffentlichen Schlüssel verwenden, um zu überprüfen, ob das Daemon-ZIP-Archiv ein Original und unverändert ist. Importieren Sie zunächst den öffentlichen Schlüssel mit GnuPG.

So importieren Sie den öffentlichen Schlüssel

1. Laden Sie den öffentlichen Schlüssel herunter.

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
$ wget $BUCKETURL/xray-daemon/aws-xray.gpg
```

2. Importieren Sie den öffentlichen Schlüssel in Ihren Schlüsselbund.

```
$ 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
```

Verwenden Sie den importierten Schlüssel, um die Signatur des Daemon-ZIP-Archivs zu überprüfen.

So überprüfen Sie die Signatur eines Archivs

1. Laden Sie das Archiv und die Signaturdatei herunter.

```
$ 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. Führen Sie `gpg --verify` aus, um die Signatur zu überprüfen.

```
$ 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
```

Beachten Sie die Warnung zu vertrauenswürdigen Inhalten. Ein Schlüssel ist nur vertrauenswürdig, wenn Sie oder eine Person Ihres Vertrauens ihn signiert hat. Das bedeutet nicht, dass die Signatur ungültig ist, sondern nur, dass Sie den öffentlichen Schlüssel nicht überprüft haben.

Ausführen des Daemons

Führen Sie den Daemon lokal über eine Befehlszeile aus. Verwenden Sie die Option `-o` zur Ausführung im lokalen Modus und `-n` zum Festlegen der Region.

```
~/Downloads$ ./xray -o -n us-east-2
```

Detaillierte plattformspezifische Anweisungen finden Sie in den folgenden Themen:

- Linux (lokal) — Den X-Ray-Daemon unter Linux ausführen
- Windows (lokal) — Den X-Ray-Daemon unter Windows ausführen
- Elastic Beanstalk — Den X-Ray-Daemon ausführen auf AWS Elastic Beanstalk
- Amazon EC2 — Den X-Ray-Daemon auf Amazon ausführen EC2
- Amazon ECS — Den X-Ray-Daemon auf Amazon ECS ausführen

Sie können mithilfe von Befehlszeilenoptionen oder einer Konfigurationsdatei das Verhalten des Daemons anpassen. Details dazu finden Sie unter Konfiguration des AWS X-Ray Daemons.

Dem Daemon die Erlaubnis geben, Daten an X-Ray zu senden

Der X-Ray-Daemon verwendet das AWS SDK, um Trace-Daten auf X-Ray hochzuladen, und benötigt dazu AWS Zugangsdaten mit entsprechender Genehmigung.

Bei Amazon EC2 verwendet der Daemon automatisch die Instance-Profilrolle der Instance. Informationen zu den Anmeldeinformationen, die für die lokale Ausführung des Daemons erforderlich sind, finden Sie unter Lokales Ausführen Ihrer Anwendung.

Wenn Sie an mehreren Orten (Anmeldeinformationsdatei, Instance-Profil oder Umgebungsvariablen) Anmeldeinformationen angeben, bestimmt die SDK-Anbieterkette, welche Anmeldeinformationen verwendet werden. Weitere Informationen zur Bereitstellung von Anmeldeinformationen für das SDK finden Sie unter Spezifying Credentials im AWS SDK for Go Developer Guide.

Die IAM-Rolle oder der Benutzer, zu der bzw. dem die Anmeldeinformationen des Daemons gehören, benötigt die Berechtigung zum Schreiben von Daten in den Service in Ihrem Namen.

- Um den Daemon auf Amazon zu verwenden EC2, erstellen Sie eine neue Instance-Profilrolle oder fügen Sie die verwaltete Richtlinie zu einer vorhandenen hinzu.
- Um den Daemon auf Elastic Beanstalk zu verwenden, fügen Sie die verwaltete Richtlinie der Standard-Instance-Profilrolle von Elastic Beanstalk hinzu.
- Informationen zum lokalen Ausführen des Daemons finden Sie unter Lokales Ausführen Ihrer Anwendung.

Weitere Informationen finden Sie unter Identitäts- und Zugriffsmanagement für AWS X-Ray.

X-Ray-Daemon-Protokolle

Der Daemon gibt Informationen über seine aktuelle Konfiguration und die Segmente aus, an die er sendet. 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)
```

Der Daemon gibt Protokolle standardmäßig an STDOUT aus. Wenn Sie den Daemon im Hintergrund ausführen, verwenden Sie zum Festlegen des Protokolldateipfades die Befehlszeilenoption `--log-file` oder eine Konfigurationsdatei. Sie können auch die Protokollierungsebene festlegen und die Protokoll-Rotation deaktivieren. Detaillierte Anweisungen finden Sie unter Konfiguration des AWS X-Ray Daemons.

Auf Elastic Beanstalk legt die Plattform den Speicherort der Daemon-Logs fest. Details dazu finden Sie unter Den X-Ray-Daemon ausführen auf AWS Elastic Beanstalk.

Konfiguration des AWS X-Ray Daemons

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können Befehlszeilenoptionen oder eine Konfigurationsdatei verwenden, um das Verhalten des X-Ray-Daemons anzupassen. Die meisten Optionen sind bei beiden Methoden verfügbar. Manche sind jedoch nur in Form von Konfigurationsdateien oder nur in der Befehlszeile verfügbar.

Um zu beginnen, müssen Sie nur die Option `-n` or kennen, mit der Sie die Region festlegen`--region`, die der Daemon verwendet, um Trace-Daten an X-Ray zu senden.

```
~/xray-daemon$ ./xray -n us-east-2
```

Wenn Sie den Daemon lokal ausführen, also nicht bei Amazon, können Sie die `-o` Option hinzufügen EC2, die Überprüfung der Anmeldeinformationen für das Instance-Profil zu überspringen, sodass der Daemon schneller bereit ist.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Mithilfe der restlichen Befehlszeilenoptionen können Sie die Protokollierung konfigurieren, einen anderen Port abhören, die vom Daemon nutzbare Speichermenge begrenzen oder eine Rolle annehmen, um Ablaufverfolgungsdaten an ein anderes Konto zu senden.

Sie können eine Konfigurationsdatei an den Daemon übergeben, um auf erweiterte Konfigurationsoptionen zuzugreifen und Dinge wie die Anzahl gleichzeitiger Aufrufe von X-Ray zu begrenzen, die Protokollrotation zu deaktivieren und Traffic an einen Proxy zu senden.

Sections

Unterstützte Umgebungsvariablen

Der X-Ray-Daemon unterstützt die folgenden Umgebungsvariablen:

- `AWS_REGION`— Gibt den AWS-Regiondes X-Ray-Dienstendpunkts an.
- `HTTPS_PROXY`— Gibt eine Proxyadresse an, über die der Daemon Segmente hochladen soll. Dies können entweder die DNS-Domänennamen oder die IP-Adressen und Portnummern sein, die von Ihren Proxy-Servern verwendet werden.

Verwenden von Befehlszeilenoptionen

Übergeben Sie diese Optionen an den Daemon, wenn Sie ihn lokal oder mit einem Benutzerdatenskript ausführen.

Befehlszeilenoptionen

- `-b`, `--bind` — Hört auf Segmentdokumente an einem anderen UDP-Port.

  ```
  --bind "127.0.0.1:3000"
  ```

  Standard —`2000`.
- `-t`, `--bind-tcp` — Warten Sie auf Anrufe an den X-Ray-Dienst an einem anderen TCP-Port.

  ```
  -bind-tcp "127.0.0.1:3000"
  ```

  Standard —`2000`.
- `-c`, `--config` — Lädt eine Konfigurationsdatei aus dem angegebenen Pfad.

  ```
  --config "/home/ec2-user/xray-daemon.yaml"
  ```
- `-f`, `--log-file` — Gibt Protokolle im angegebenen Dateipfad aus.

  ```
  --log-file "/var/log/xray-daemon.log"
  ```
- `-l`, `--log-level` — Protokollebene, von der umfangreichsten bis zur geringsten Ausführlichkeit: dev, debug, info, warn, error, prod.

  ```
  --log-level warn
  ```

  Standard — `prod`
- `-m`, `--buffer-memory` — Ändert die Speichermenge in MB, die Puffer verwenden können (mindestens 3).

  ```
  --buffer-memory 50
  ```

  Standard — 1% des verfügbaren Speichers.
- `-o`, `--local-mode` — Suchen Sie nicht nach EC2 Instanz-Metadaten.
- `-r`, `--role-arn` — Nehmen Sie die angegebene IAM-Rolle an, um Segmente auf ein anderes Konto hochzuladen.

  ```
  --role-arn "arn:aws:iam::123456789012:role/xray-cross-account"
  ```

- `-a`, `--resource-arn` — Amazon-Ressourcenname (ARN) der AWS Ressource, auf der der Daemon ausgeführt wird.
- `-p`, `--proxy-address` — Laden Sie Segmente AWS X-Ray über einen Proxy hoch. Das Protokoll des Proxyservers muss angegeben werden.

```
--proxy-address "http://192.0.2.0:3000"
```

- `-n`, `--region` — Sendet Segmente an den X-Ray-Dienst in einer bestimmten Region.
- `-v`, `--version` — AWS X-Ray Daemon-Version anzeigen.
- `-h`, `--help` — Zeigt den Hilfebildschirm an.

Verwendung einer Konfigurationsdatei

Sie können auch eine Datei im YAML-Format zur Konfiguration des Daemons verwenden. Übergeben Sie die Konfigurationsdatei mit der `-c`-Option an den Daemon.

```
~$ ./xray -c ~/xray-daemon.yaml
```

Konfigurationsdateioptionen

- `TotalBufferSizeMB`— Maximale Puffergröße in MB (mindestens 3). Wählen Sie 0, um 1 % des Host-Speichers zu verwenden.
- `Concurrency`— Maximale Anzahl gleichzeitiger Aufrufe AWS X-Ray zum Hochladen von Segmentdokumenten.
- `Region`— Senden Sie Segmente an den AWS X-Ray Service in einer bestimmten Region.
- `Socket`— Konfiguriert die Bindung des Daemons.
  - `UDPAddress`— Ändert den Port, auf dem der Daemon lauscht.
  - `TCPAddress`— Achten Sie auf Anrufe an den X-Ray-Dienst an einem anderen TCP-Port.
- `Logging`— Konfigurieren Sie das Protokollierungsverhalten.
  - `LogRotation`— Auf einstellen, `false` um die Protokollrotation zu deaktivieren.
  - `LogLevel`— Ändert die Protokollebene von der ausführlichsten zur geringsten:`dev`,`debug`, `info` oder`prod`,, `warn`,`error`,`prod`. Die Standardeinstellung ist`prod`, was äquivalent zu `info` ist.
  - `LogPath`— Gibt Protokolle im angegebenen Dateipfad aus.

- `LocalMode`— Auf setzen, `true` um die Suche nach EC2 Instanz-Metadaten zu überspringen.
- `ResourceARN`— Amazon-Ressourcenname (ARN) der AWS Ressource, auf der der Daemon ausgeführt wird.
- `RoleARN`— Nehmen Sie die angegebene IAM-Rolle an, um Segmente auf ein anderes Konto hochzuladen.
- `ProxyAddress`— Laden Sie Segmente AWS X-Ray über einen Proxy hoch.
- `Endpoint`— Ändert den X-Ray-Dienstendpunkt, an den der Daemon Segmentdokumente sendet.
- `NoVerifySSL`— Deaktiviert die Überprüfung des TLS-Zertifikats.
- `Version`— Version des Formats der Daemon-Konfigurationsdatei. Die Version des Dateiformats ist ein Pflichtfeld.

Example Xray-daemon.yaml

In dieser Konfigurationsdatei wird der Listening-Port des Daemons in 3000 geändert. Dazu werden die Prüfungen auf Instance-Metadaten deaktiviert, eine Rolle für den Upload von Segmenten eingerichtet und die Optionen für Region und Protokollierung geändert.

```
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
```

Lokales Ausführen des X-Ray-Daemons

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon
> Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu

OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können den AWS X-Ray Daemon lokal unter Linux, macOS, Windows oder in einem Docker-Container ausführen. Führen Sie den Daemon aus, um Trace-Daten an X-Ray weiterzuleiten, wenn Sie Ihre instrumentierte Anwendung entwickeln und testen. Laden Sie den Daemon herunter und extrahieren Sie ihn. Diesbezügliche Anweisungen finden Sie hier.

Wenn der Daemon lokal ausgeführt wird, kann er Anmeldeinformationen aus einer AWS SDK-Anmeldeinformationsdatei (`.aws/credentials`in Ihrem Benutzerverzeichnis) oder aus Umgebungsvariablen lesen. Weitere Informationen finden Sie unter Dem Daemon die Erlaubnis geben, Daten an X-Ray zu senden.

Der Daemon lauscht an Port 2000 auf Daten. Sie können den Port und andere Optionen mithilfe einer Konfigurationsdatei und von Befehlszeilenoptionen ändern. Weitere Informationen finden Sie unter Konfiguration des AWS X-Ray Daemons.

Den X-Ray-Daemon unter Linux ausführen

Sie können die ausführbare Datei des Daemons an der Befehlszeile ausführen. Verwenden Sie die Option `-o` zur Ausführung im lokalen Modus und `-n` zum Festlegen der Region.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Zum Ausführen des Daemons im Hintergrund verwenden Sie &.

```
~/xray-daemon$ ./xray -o -n us-east-2 &
```

Beenden Sie einen Daemonprozess, der im Hintergrund ausgeführt wird, mit `pkill`.

```
~$ pkill xray
```

Den X-Ray-Daemon in einem Docker-Container ausführen

Um den Daemon lokal in einem Docker-Container auszuführen, speichern Sie den folgenden Text in einer Datei mit dem Namen `Dockerfile`. Laden Sie das vollständige Beispielbild auf Amazon ECR herunter. Weitere Informationen finden Sie unter Den Daemon herunterladen.

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
```

Entwickeln Sie das Container-Image mit `docker build`.

```
~/xray-daemon$ docker build -t xray-daemon .
```

Führen Sie das Abbild in einem Container mit `docker run` aus.

```
~/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
```

Dieser Befehl verwendet die folgenden Optionen:

- `--attach STDOUT`— Zeigt die Ausgabe des Daemons im Terminal an.
- `-v ~/.aws/:/root/.aws/:ro`— Gewähren Sie dem Container nur Lesezugriff auf das `.aws` Verzeichnis, damit er Ihre AWS SDK-Anmeldeinformationen lesen kann.
- `AWS_REGION=`*`us-east-2`*— Setzt die `AWS_REGION` Umgebungsvariable, um dem Daemon mitzuteilen, welche Region er verwenden soll.
- `--net=host`— Hängt den Container an das `host` Netzwerk an. Container im Hostnetzwerk können miteinander kommunizieren, ohne Ports zu veröffentlichen.
- `-p 2000:2000/udp`— Ordnen Sie den UDP-Port 2000 auf Ihrem Computer demselben Port auf dem Container zu. Dies ist für die Kommunikation von Containern in demselben Netzwerk nicht erforderlich. Sie können mit ihr aber Segmente über die Befehlszeile oder Anwendung, die nicht in Docker ausgeführt wird, zum Daemon senden.

- `--name xray-daemon`— Benennen Sie den Container, `xray-daemon` anstatt einen zufälligen Namen zu generieren.
- `-o`(nach dem Image-Namen) — Hängen Sie die `-o` Option an den Einstiegspunkt an, der den Daemon innerhalb des Containers ausführt. Diese Option weist den Daemon an, im lokalen Modus zu laufen, um zu verhindern, dass er versucht, EC2 Amazon-Instance-Metadaten zu lesen.

Um den Daemon zu beenden, verwenden Sie `docker stop`. Wenn Sie nach der Vornahme von Änderungen am `Dockerfile` ein neues Abbild erstellen, müssen Sie den vorhandenen Container löschen, damit Sie einen anderen mit demselben Namen erstellen können. Verwenden Sie `docker rm` zum Löschen des Containers.

```
$ docker stop xray-daemon
$ docker rm xray-daemon
```

Den X-Ray-Daemon unter Windows ausführen

Sie können die ausführbare Datei des Daemons an der Befehlszeile ausführen. Verwenden Sie die Option `-o` zur Ausführung im lokalen Modus und `-n` zum Festlegen der Region.

```
> .\xray_windows.exe -o -n us-east-2
```

Verwenden Sie ein PowerShell Skript, um einen Dienst für den Daemon zu erstellen und auszuführen.

Example PowerShell Skript — 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
```

Den X-Ray-Daemon auf OS X ausführen

Sie können die ausführbare Datei des Daemons an der Befehlszeile ausführen. Verwenden Sie die Option `-o` zur Ausführung im lokalen Modus und `-n` zum Festlegen der Region.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2
```

Zum Ausführen des Daemons im Hintergrund verwenden Sie &.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2 &
```

Verwenden Sie nohup, um zu verhindern, dass der Daemon vorzeitig beendet wird, wenn das Terminal geschlossen wird.

```
~/xray-daemon$ nohup ./xray_mac &
```

Den X-Ray-Daemon ausführen auf AWS Elastic Beanstalk

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Um Trace-Daten von Ihrer Anwendung an weiterzuleiten AWS X-Ray, können Sie den X-Ray-Daemon auf den Amazon-Instances Ihrer Elastic Beanstalk Beanstalk-Umgebung ausführen. EC2 Eine Liste der unterstützten Plattformen finden Sie unter Configuring AWS X-Ray Debugging im Developer Guide.AWS Elastic Beanstalk

> **Note**
>
> Der Daemon verwendet das Instance-Profil Ihrer Umgebung für Berechtigungen. Anweisungen zum Hinzufügen von Berechtigungen zum Elastic Beanstalk Beanstalk-Instance-Profil finden Sie unter. Dem Daemon die Erlaubnis geben, Daten an X-Ray zu senden

Elastic Beanstalk-Plattformen bieten eine Konfigurationsoption, die Sie so einrichten können, dass der Daemon automatisch ausgeführt wird. Sie können den Daemon in einer Konfigurationsdatei in Ihrem Quellcode oder durch Auswahl einer Option in der Elastic Beanstalk Beanstalk-Konsole aktivieren. Wenn Sie die Konfigurationsoption aktivieren, wird der Daemon in der Instance installiert und als Service ausgeführt.

Die auf Elastic Beanstalk-Plattformen enthaltene Version ist möglicherweise nicht die neueste Version. Dem Thema "Unterstützte Plattformen" können Sie entnehmen, welche Version des Daemons für Ihre Plattformkonfiguration verfügbar ist.

Elastic Beanstalk stellt den X-Ray-Daemon auf der Multicontainer Docker (Amazon ECS) -Plattform nicht bereit.

Verwenden der Elastic Beanstalk X-Ray-Integration zum Ausführen des X-Ray-Daemons

Verwenden Sie die Konsole, um die X-Ray-Integration zu aktivieren, oder konfigurieren Sie sie in Ihrem Anwendungsquellcode mit einer Konfigurationsdatei.

So aktivieren Sie den X-Ray-Daemon in der Elastic Beanstalk Beanstalk-Konsole

1. In der Elastic-Beanstalk-Konsole öffnen.
2. Navigieren Sie zur Managementkonsole für Ihre Umgebung.
3. Wählen Sie Konfiguration.
4. Wählen Sie Software Settings (Softwareeinstellungen).

5. Wählen Sie für X-Ray daemon Enabled.
6. Wählen Sie Anwenden aus.

Sie können eine Konfigurationsdatei in Ihren Quellcode aufnehmen, um die Konfiguration zwischen verschiedenen Umgebungen übertragbar zu gestalten.

Example.ebextensions/xray-daemon.config

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

Elastic Beanstalk übergibt eine Konfigurationsdatei an den Daemon und gibt die Protokolle an einem Standardspeicherort aus.

Auf Windows Server-Plattformen

- Konfigurationsdatei — `C:\Program Files\Amazon\XRay\cfg.yaml`
- Protokolle — `c:\Program Files\Amazon\XRay\logs\xray-service.log`

Auf Linux-Plattformen

- Konfigurationsdatei — `/etc/amazon/xray/cfg.yaml`
- Protokolle — `/var/log/xray/xray.log`

Elastic Beanstalk bietet Tools zum Abrufen von Instance-Logs über die Befehlszeile AWS-Managementkonsole oder. Sie können Elastic Beanstalk anweisen, die X-Ray-Daemon-Logs einzubeziehen, indem Sie eine Aufgabe mit einer Konfigurationsdatei hinzufügen.

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
```

Weitere Informationen finden Sie im AWS Elastic Beanstalk Developer Guide unter Logs aus den EC2 Amazon-Instances Ihrer Elastic Beanstalk Beanstalk-Umgebung anzeigen.

Manuelles Herunterladen und Ausführen des X-Ray-Daemons (fortgeschritten)

Wenn der X-Ray-Daemon für Ihre Plattformkonfiguration nicht verfügbar ist, können Sie ihn von Amazon S3 herunterladen und mit einer Konfigurationsdatei ausführen.

Verwenden Sie eine Elastic Beanstalk Beanstalk-Konfigurationsdatei, um den Daemon herunterzuladen und auszuführen.

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
```

Diese Beispiele fügen auch die Log-Datei des Daemons zur Elastic Beanstalk-Task für die Tail-Logs hinzu, sodass sie enthalten ist, wenn Sie Logs mit der Konsole oder der Elastic Beanstalk Command Line Interface (EB CLI) anfordern.

Den X-Ray-Daemon auf Amazon ausführen EC2

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können den X-Ray-Daemon auf den folgenden Betriebssystemen bei Amazon EC2 ausführen:

- Amazon Linux
- Ubuntu
- Windows Server (2012 R2 und neuer)

Verwenden Sie ein Instanzprofil, um dem Daemon die Erlaubnis zu erteilen, Trace-Daten auf X-Ray hochzuladen. Weitere Informationen finden Sie unter Dem Daemon die Erlaubnis geben, Daten an X-Ray zu senden.

Verwenden Sie ein Benutzerdatenskript, um den Daemon automatisch auszuführen, wenn Sie die Instance starten.

Example Benutzerdatenskript – 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 Benutzerdatenskript – 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>
```

Den X-Ray-Daemon auf Amazon ECS ausführen

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Erstellen Sie in Amazon ECS ein Docker-Image, das den X-Ray-Daemon ausführt, laden Sie es in ein Docker-Image-Repository hoch und stellen Sie es dann in Ihrem Amazon ECS-Cluster bereit. Sie können Ihrer Anwendung über Port-Zuordnungen und Netzwerkmodus-Einstellungen in Ihrer Aufgabendefinitionsdatei die Kommunikation mit dem Daemon-Container ermöglichen.

Verwenden des offiziellen -Docker-Image

X-Ray stellt ein Docker-Container-Image auf Amazon ECR bereit, das Sie zusammen mit Ihrer Anwendung bereitstellen können. Weitere Informationen finden Sie unter Den Daemon herunterladen.

Example Aufgabendefinition

```
{
  "name": "xray-daemon",
  "image": "amazon/aws-xray-daemon",
  "cpu": 32,
  "memoryReservation": 256,
  "portMappings" : [
      {
          "hostPort": 0,
          "containerPort": 2000,
          "protocol": "udp"
      }
   ]
}
```

Entwickeln und Erstellen eines Docker-Images

Bei einer benutzerdefinierten Konfiguration müssen Sie möglicherweise ein eigenes Docker-Image definieren.

Fügen Sie verwaltete Richtlinien zu Ihrer Aufgabenrolle hinzu, um dem Daemon die Erlaubnis zu erteilen, Trace-Daten auf X-Ray hochzuladen. Weitere Informationen finden Sie unter Dem Daemon die Erlaubnis geben, Daten an X-Ray zu senden.

Erstellen Sie mit einer der folgenden Dockerfiles ein Abbild, mit dem der Daemon ausführt wird.

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**
>
> Die Flags `-t` und `-b` sind erforderlich, um eine Bindungsadresse anzugeben, die den Loopback einer Multi-Container-Umgebung überwacht.

Example Dockerfile — Ubuntu

Für Debian-Derivate müssen Sie auch Zertifikate installieren, die von einer Zertifizierungsstelle (Certificate Authority, CA) ausgegeben wurden, um Probleme beim Herunterladen des Installationsprogramms zu vermeiden.

```
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
```

In Ihrer Aufgabendefinition hängt die Konfiguration von dem Netzwerkmodus ab, den Sie tatsächlich nutzen. Brückennetzwerke sind die Standardeinstellung und können in Ihrer Standard-VPC verwendet werden. Stellen Sie in einem Bridge-Netzwerk die `AWS_XRAY_DAEMON_ADDRESS` Umgebungsvariable so ein, dass sie dem X-Ray SDK mitteilt, auf welchen Container-Port verwiesen werden soll, und legen Sie den Host-Port fest. Sie können beispielsweise UDP-Port 2000 veröffentlichen und einen Link von Ihrem Anwendungs-Container zum Daemon-Container erstellen.

Example Aufgabendefinition

```
    {
      "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"
      ]
    }
```

Wenn Sie Ihren Cluster im privaten Subnetz einer VPC ausführen, können Sie Ihren Containern mit dem awsvpc-Netzwerkmodus einer Elastic Network-Schnittstelle (ENI) zuweisen. Auf diese Weise können Sie die Nutzung von Links vermeiden. Lassen Sie den Host-Port in den Port-Zuordnungen, dem Link und der Umgebungsvariable `AWS_XRAY_DAEMON_ADDRESS` weg.

Example VPC-Aufgabendefinition

```
{
    "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
                }
```

```
            ]
        }
    ]
}
```

Befehlszeilenoptionen in der Amazon ECS-Konsole konfigurieren

Befehlszeilenoptionen überschreiben Konflikte in der Konfigurationsdatei Ihres Abbildes. Befehlszeilenoptionen werden normalerweise für lokale Tests verwendet, können aber auch aus praktischen Gründen beim Festlegen von Umgebungsvariablen oder zur Steuerung des Startvorgangs verwendet werden.

Durch Hinzufügen von Befehlszeilenoptionen aktualisieren Sie den Docker `CMD`, der an den Container übergeben wird. Weitere Informationen finden Sie in der Referenz zu docker run.

So legen Sie eine Befehlszeilenoption fest

1. Öffnen Sie die klassische Amazon ECS-Konsole unter https://console.aws.amazon.com/ecs/.
2. Wählen Sie auf der Navigationsleiste die Region aus, in der Ihre Aufgabendefinition enthalten ist.
3. Wählen Sie im Navigationsbereich Task Definitions aus.
4. Aktivieren Sie auf der Seite Task Definitions das Kontrollkästchen links neben der Aufgabendefinition, die Sie überarbeiten möchten, und wählen Sie dann Create new revision aus.
5. Wählen Sie auf der Seite Create new revision of Task Definition (Neue Revision der Aufgabendefinition erstellen) den Container aus.
6. Fügen Sie im Abschnitt ENVIRONMENT Ihre durch Komma getrennte Liste von Befehlszeilenoptionen zum Feld Command hinzu.
7. Wählen Sie Aktualisieren aus.
8. Überprüfen Sie die Informationen und wählen Sie dann Create aus.

Das folgende Beispiel zeigt, wie eine durch Kommas getrennte Befehlszeilenoption für die Option `RoleARN` geschrieben wird. Die `RoleARN` Option setzt die angegebene IAM-Rolle voraus, um Segmente auf ein anderes Konto hochzuladen.

Example

```
--role-arn, arn:aws:iam::123456789012:role/xray-cross-account
```

Weitere Informationen zu den verfügbaren Befehlszeilenoptionen in X-Ray finden Sie unter Konfiguration des AWS X-Ray Daemons.

Integration AWS X-Ray mit anderen AWS-Services

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Viele AWS-Services bieten unterschiedliche Stufen der X-Ray-Integration, darunter Sampling und Hinzufügen von Headern zu eingehenden Anfragen, Ausführung des X-Ray-Daemons und automatisches Senden von Trace-Daten an X-Ray. Die Integration mit X-Ray kann Folgendes beinhalten:

- Aktive Instrumentierung — Eingehende Anfragen für Proben und Instrumente
- Passive Instrumentierung — Anfragen zu Instrumenten, die von einem anderen Dienst gesampelt wurden
- Anforderungsverfolgung — Fügt allen eingehenden Anfragen einen Tracing-Header hinzu und leitet ihn anschließend weiter
- Tooling — Führt den X-Ray-Daemon aus, um Segmente vom X-Ray SDK zu empfangen

> **Note**
>
> Das X-Ray SDKs enthält Plugins für eine zusätzliche Integration mit AWS-Services. Sie können beispielsweise das Elastic Beanstalk-Plug-In X-Ray SDK for Java verwenden, um Informationen über die Elastic Beanstalk Beanstalk-Umgebung hinzuzufügen, in der Ihre Anwendung ausgeführt wird, einschließlich des Umgebungsnamens und der ID.

Hier sind einige Beispiele dafür AWS-Services , die in X-Ray integriert sind:

- AWS Distro for OpenTelemetry (ADOT) — Mit ADOT können Ingenieure ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Überwachungslösungen senden, darunter Amazon CloudWatch AWS X-Ray, Amazon Service und Amazon Managed OpenSearch Service for Prometheus.
- AWS Lambda— Aktive und passive Instrumentierung eingehender Anfragen zu allen Laufzeiten. AWS Lambda fügt Ihrer Trace-Map zwei Knoten hinzu, einen für den AWS Lambda Service und einen für die Funktion. Wenn Sie die Instrumentierung aktivieren, wird AWS Lambda auch der X-Ray-Daemon auf Java- und Node.js Runtimes zur Verwendung mit dem X-Ray SDK ausgeführt.
- Amazon API Gateway — Aktive und passive Instrumentierung. API Gateway verwendet Stichprobenregeln, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen, und fügt Ihrer Service-Map einen Knoten für die Gateway-Phase hinzu.
- AWS Elastic Beanstalk— Werkzeuge. Elastic Beanstalk enthält den X-Ray-Daemon auf den folgenden Plattformen:
  - Java SE — 2.3.0 und spätere Konfigurationen
  - Tomcat — 2.4.0 und spätere Konfigurationen
  - Node.js — 3.2.0 und spätere Konfigurationen
  - Windows Server — Alle Konfigurationen außer Windows Server Core, die nach dem 9. Dezember 2016 veröffentlicht wurden

  Sie können die Elastic Beanstalk-Konsole verwenden, um Elastic Beanstalk anzuweisen, den Daemon auf diesen Plattformen auszuführen, oder Sie können die `XRayEnabled` Option im Namespace verwenden. `aws:elasticbeanstalk:xray`
- Elastic Load Balancing — Ablaufverfolgung von Anfragen auf Application Load Balancers. Der Application Load Balancer fügt die Trace-ID dem Anforderungsheader hinzu, bevor er ihn an eine Zielgruppe sendet.
- Amazon EventBridge — Passive Instrumentierung. Wenn ein Dienst, der Ereignisse veröffentlicht, mit dem X-Ray SDK instrumentiert EventBridge ist, erhalten die Ereignisziele den Tracing-Header und können weiterhin die ursprüngliche Trace-ID weitergeben.
- Amazon Simple Notification Service — Passive Instrumentierung. Wenn ein Amazon SNS SNS-Publisher seinen Kunden mit dem X-Ray SDK verfolgt, können Abonnenten den Tracing-Header abrufen und den ursprünglichen Trace vom Herausgeber mit derselben Trace-ID weiterleiten.
- Amazon Simple Queue Service — Passive Instrumentierung. Wenn ein Service Anfragen mithilfe des X-Ray-SDK verfolgt, kann Amazon SQS den Tracing-Header senden und den ursprünglichen Trace mit einer konsistenten Trace-ID weiterhin vom Absender an den Verbraucher weitergeben.

- Amazon Bedrock AgentCore — AgentCore unterstützt verteiltes Tracing durch X-Ray-Integration, sodass Sie Anfragen verfolgen können, während sie durch Ihre Agentenanwendungen fließen. Wenn Sie Observability für Ihre AgentCore Ressourcen aktivieren, können Sie den Trace-Kontext über Servicegrenzen hinweg weitergeben und sich so einen Überblick über die Leistung Ihrer KI-Agenten und -Tools verschaffen.

Wählen Sie aus den folgenden Themen, um sich mit allen integrierten Funktionen vertraut zu machen. AWS-Services

Themen

Amazon Bedrock AgentCore und AWS X-Ray

Amazon Bedrock AgentCore lässt sich integrieren AWS X-Ray , um verteilte Tracking-Funktionen für Ihre KI-Agenten und -Tools bereitzustellen. Diese Integration ermöglicht es Ihnen, Anfragen zu

verfolgen, während sie Ihre Agentenanwendungen durchlaufen, was Ihnen hilft, Leistungsengpässe zu identifizieren und Probleme zu beheben.

AgentCore unterstützt verteiltes Tracing durch X-Ray-Integration, sodass Sie die Leistung Ihrer KI-Agenten und -Tools überwachen können. Wenn Sie Observability für Ihre AgentCore Ressourcen aktivieren, können Sie den Trace-Kontext über Servicegrenzen hinweg weitergeben und sich einen Überblick darüber verschaffen, wie Ihre Agenten mit anderen Diensten interagieren. AWS Weitere Informationen finden Sie unter Amazon Bedrock AgentCore.

AgentCore unterstützt die folgenden X-Ray-Funktionen:

- Weitergabe des Trace-Kontextes an nachgeschaltete Dienste
- Benutzerdefinierte Instrumentierung mit dem AWS Distro for OpenTelemetry (ADOT) SDK

X-Ray einrichten mit AgentCore

Um X-Ray mit verwenden zu können AgentCore, müssen Sie die CloudWatch Transaktionssuche in Ihrem AWS Konto aktivieren. Dies ist ein einmaliges Setup, mit dem Trace-Daten AgentCore an X-Ray gesendet werden können. Weitere Informationen finden Sie unter Transaktionssuche aktivieren.

Weitere Informationen zum Einrichten von Observability für AgentCore finden Sie unter Observability zu Ihrem Amazon AgentCore Bedrock-Agenten oder -Tool hinzufügen.

Verwenden von Trace-Headern mit AgentCore

AgentCore unterstützt das X-Ray-Trace-Header-Format für verteiltes Tracing. Sie können den `X-Amzn-Trace-Id` Header in Ihre Anfragen aufnehmen, AgentCore um den Trace-Kontext über Dienstgrenzen hinweg aufrechtzuerhalten.

Amazon Elastic Compute Cloud und AWS X-Ray

Sie können den X-Ray-Daemon auf einer EC2 Amazon-Instance mit einem Benutzerdatenskript installieren und ausführen. Detaillierte Anweisungen finden Sie unter Den X-Ray-Daemon auf Amazon ausführen EC2.

Verwenden Sie ein Instanzprofil, um dem Daemon die Erlaubnis zu erteilen, Trace-Daten auf X-Ray hochzuladen. Weitere Informationen finden Sie unter Dem Daemon die Erlaubnis geben, Daten an X-Ray zu senden.

Amazon SNS und AWS X-Ray

Sie können Amazon Simple Notification Service (Amazon SNS) verwenden AWS X-Ray , um Anfragen zu verfolgen und zu analysieren, während sie über Ihre SNS-Themen an Ihre von SNS unterstützten Abonnementdienste weitergeleitet werden. Verwenden Sie X-Ray Tracing mit Amazon SNS, um Latenzen in Ihren Nachrichten und deren Backend-Services zu analysieren, z. B. wie lange eine Anfrage in einem Thema verweilt und wie lange es dauert, bis die Nachricht an die einzelnen Abonnements des Themas zugestellt wird. Amazon SNS unterstützt X-Ray-Tracing für Standard- und FIFO-Themen.

Wenn Sie über einen Service, der bereits mit X-Ray ausgestattet ist, zu einem Amazon SNS-Thema veröffentlichen, gibt Amazon SNS den Trace-Kontext vom Herausgeber an die Abonnenten weiter. Darüber hinaus können Sie die aktive Ablaufverfolgung aktivieren, um Segmentdaten über Ihre Amazon SNS SNS-Abonnements an X-Ray für Nachrichten zu senden, die von einem instrumentierten SNS-Client veröffentlicht wurden. Aktivieren Sie die aktive Ablaufverfolgung für ein Amazon SNS SNS-Thema mithilfe der Amazon SNS SNS-Konsole oder mithilfe der Amazon SNS SNS-API oder CLI. Weitere Informationen zur Instrumentierung Ihrer SNS-Clients finden Sie unter Instrumentierung Ihrer Anwendung.

Active Tracing von Amazon SNS konfigurieren

Sie können die Amazon SNS-Konsole oder die AWS CLI oder das SDK verwenden, um Amazon SNS Active Tracing zu konfigurieren.

Wenn Sie die Amazon SNS-Konsole verwenden, versucht Amazon SNS, die erforderlichen Berechtigungen zu erstellen, damit SNS X-Ray aufrufen kann. Der Versuch kann abgelehnt werden, wenn Sie nicht über ausreichende Berechtigungen zum Ändern der X-Ray-Ressourcenrichtlinien verfügen. Weitere Informationen zu diesen Berechtigungen finden Sie unter Identitäts- und Zugriffsmanagement in Amazon SNS und Beispielfälle für Amazon SNS SNS-Zugriffskontrolle im Amazon Simple Notification Service Developer Guide. Weitere Informationen zum Aktivieren der aktiven Ablaufverfolgung mithilfe der Amazon SNS-Konsole finden Sie unter Aktivieren der aktiven Ablaufverfolgung auf einem Amazon SNS im Amazon Simple Notification Service Developer Guide.

Wenn Sie die AWS CLI oder das SDK verwenden, um Active Tracing zu aktivieren, müssen Sie die Berechtigungen mithilfe ressourcenbasierter Richtlinien manuell konfigurieren. Wird verwendet `PutResourcePolicy`, um X-Ray mit der erforderlichen ressourcenbasierten Richtlinie zu konfigurieren, damit Amazon SNS Traces an X-Ray senden kann.

Example Beispiel für eine ressourcenbasierte X-Ray-Richtlinie für Amazon SNS Active Tracing

Dieses Beispielrichtliniendokument spezifiziert die Berechtigungen, die Amazon SNS benötigt, um Trace-Daten an X-Ray zu senden:

```
{
    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"
          }
        }
      }
    ]
  }
```

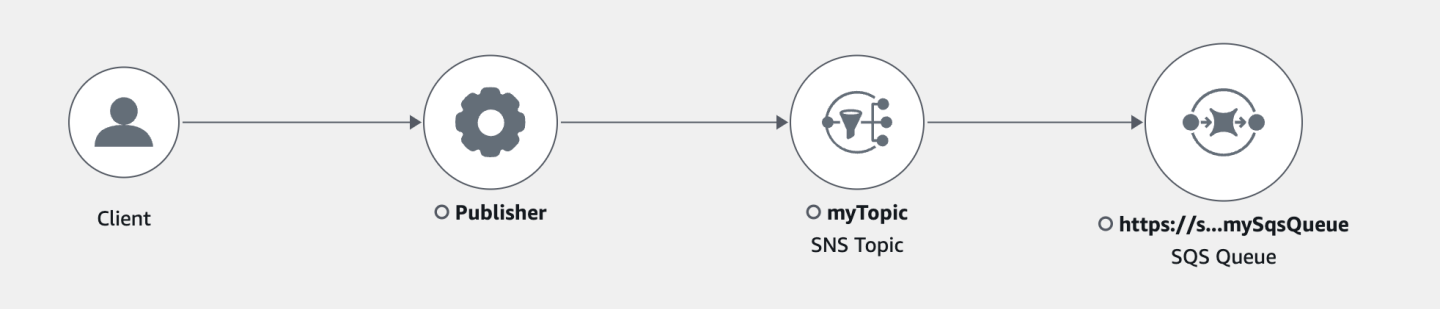
Verwenden Sie die CLI, um eine ressourcenbasierte Richtlinie zu erstellen, die Amazon SNS Berechtigungen zum Senden von Trace-Daten an X-Ray erteilt:

```
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" } } } ] }'
```

Um diese Beispiele zu verwenden, ersetzen Sie*partition*, *regionaccount-id*, und *topic-name* durch Ihre spezifische AWS Partition, Region, Konto-ID und Ihren Amazon SNS SNS-Themennamen. Um allen Amazon SNS SNS-Themen die Erlaubnis zu geben, Trace-Daten an X-Ray zu senden, ersetzen Sie den Themennamen durch. *

Amazon SNS SNS-Publisher- und Abonnenten-Traces in der X-Ray-Konsole anzeigen

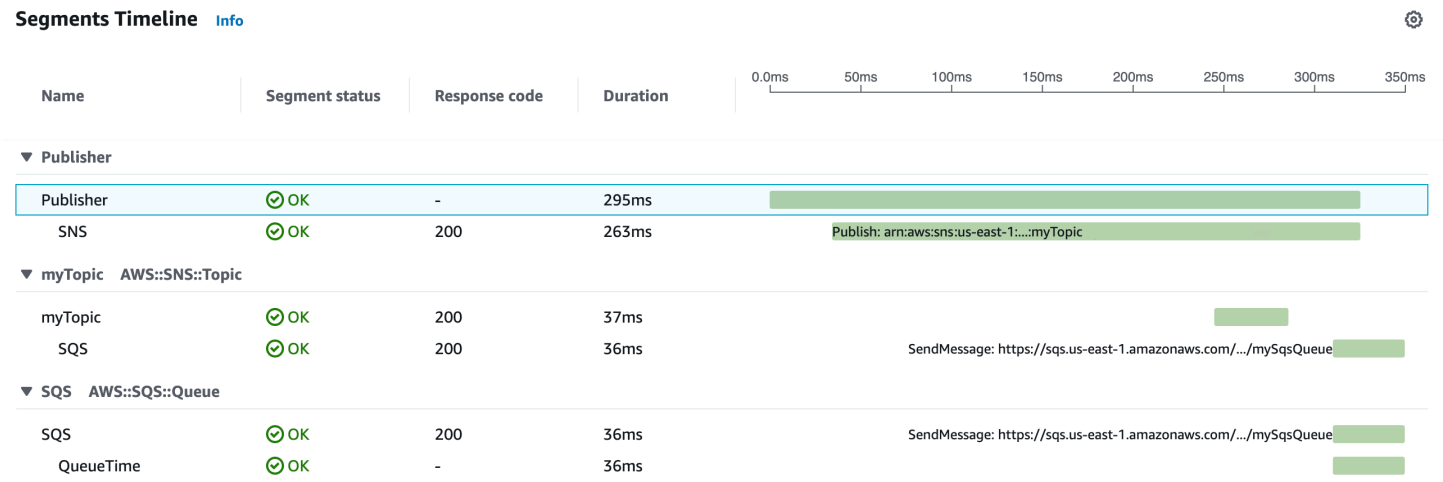
Verwenden Sie die X-Ray-Konsole, um eine Ablaufverfolgungsübersicht und Ablaufverfolgungsdetails anzuzeigen, die eine verbundene Ansicht der Amazon SNS SNS-Publisher und -Abonnenten anzeigen. Wenn das aktive Tracing von Amazon SNS für ein Thema aktiviert ist, werden in der X-Ray-Trace-Map und der Trace-Details-Map verbundene Knoten für Amazon SNS SNS-Publisher, das Amazon SNS SNS-Thema und nachgeschaltete Abonnenten angezeigt:



Nachdem Sie einen Trace ausgewählt haben, der sich über einen Amazon SNS SNS-Publisher und -Abonnenten erstreckt, werden auf der Seite mit den X-Ray-Trace-Details eine Ablaufverfolgungsdetailübersicht und eine Segment-Timeline angezeigt.

Example Beispiel-Timeline mit Amazon SNS SNS-Publisher und Abonnent

Dieses Beispiel zeigt eine Zeitleiste mit einem Amazon SNS SNS-Publisher, der eine Nachricht an ein Amazon SNS SNS-Thema sendet, die von einem Amazon SQS SQS-Abonnenten bearbeitet wird.

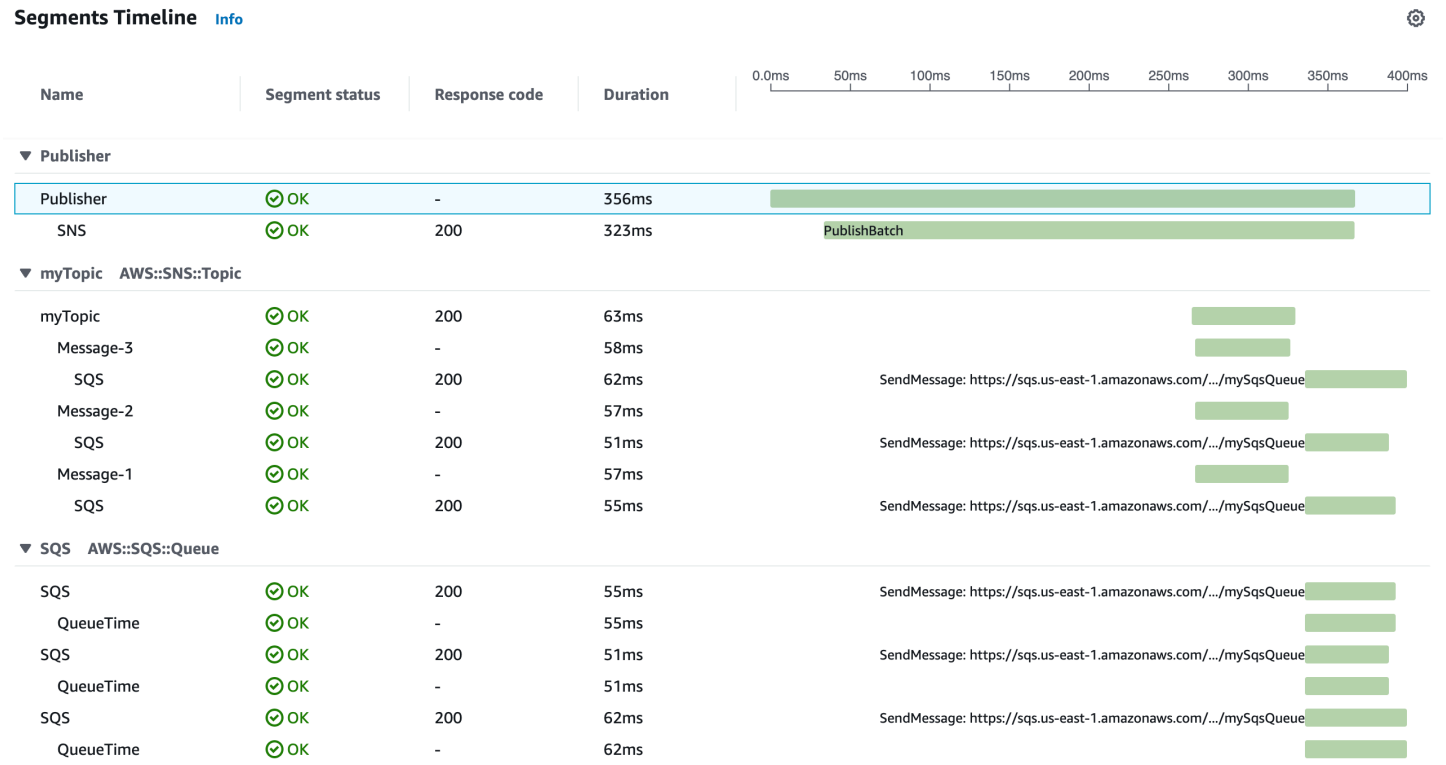


Die obige Beispielzeitleiste enthält Details zum Amazon SNS SNS-Nachrichtenfluss:

- Das SNS-Segment stellt die Round-Turn-Dauer des `Publish` API-Aufrufs vom Client dar.
- Das MyTopic-Segment stellt die Latenz der Amazon SNS SNS-Antwort auf die Veröffentlichungsanfrage dar.
- Das SQS-Subsegment stellt die Round-Trip-Zeit dar, die Amazon SNS benötigt, um die Nachricht in einer Amazon SQS SQS-Warteschlange zu veröffentlichen.
- Die Zeit zwischen dem MyTopic-Segment und dem SQS-Subsegment entspricht der Zeit, die die Nachricht im Amazon SNS SNS-System verbringt.

Example Beispiel für eine Zeitleiste mit gebündelten Amazon SNS-Nachrichten

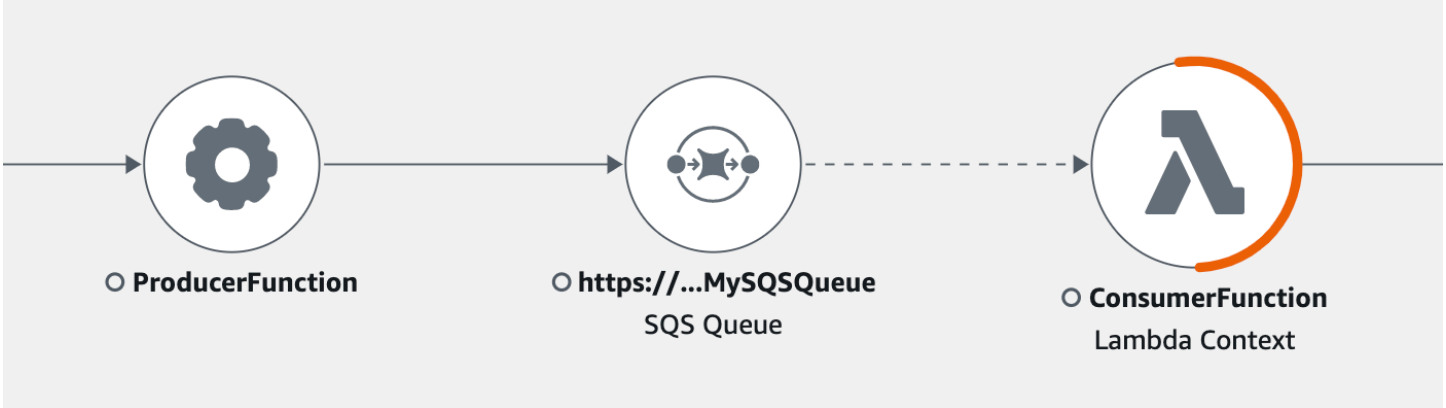
Wenn mehrere Amazon SNS SNS-Nachrichten in einem einzigen Trace gebündelt werden, zeigt die Segment-Timeline Segmente an, die jede verarbeitete Nachricht repräsentieren.



Amazon SQS und AWS X-Ray

AWS X-Ray lässt sich in Amazon Simple Queue Service (Amazon SQS) integrieren, um Nachrichten zu verfolgen, die eine Amazon SQS SQS-Warteschlange durchlaufen. Wenn ein Service Anfragen mithilfe des X-Ray-SDK verfolgt, kann Amazon SQS den Tracing-Header senden und den ursprünglichen Trace mit einer konsistenten Trace-ID weiterhin vom Absender an den Verbraucher weitergeben. Die Nachverfolgungskontinuität ermöglicht Benutzern das Nachverfolgen, Analysieren und Debuggen in allen nachgeschalteten Services.

AWS X-Ray unterstützt die Ablaufverfolgung ereignisgesteuerter Anwendungen mithilfe von Amazon SQS und. AWS Lambda Verwenden Sie die CloudWatch Konsole, um eine verbundene Ansicht jeder Anfrage zu sehen, während sie bei Amazon SQS in die Warteschlange gestellt und von einer nachgeschalteten Lambda-Funktion verarbeitet wird. Traces von Upstream-Nachrichtenproduzenten werden automatisch mit Traces von nachgeschalteten Lambda-Consumer-Knoten verknüpft, wodurch eine end-to-end Ansicht der Anwendung erstellt wird. Weitere Informationen finden Sie unter Ablaufverfolgung ereignisgesteuerter Anwendungen.



Amazon SQS unterstützt die folgende Tracing-Header-Instrumentierung:

- Standard-HTTP-Header — Das X-Ray-SDK füllt den Trace-Header automatisch als HTTP-Header auf, wenn Sie Amazon SQS über das AWS SDK aufrufen. Der Standard-Nachverfolgungs-Header wird von `X-Amzn-Trace-Id` übernommen und entspricht allen Nachrichten, die in einer `SendMessage`- oder `SendMessageBatch`-Anfrage enthalten sind. Weitere Informationen zum Standard-HTTP-Header finden Sie unter Ablaufverfolgungs-Header.
- **`AWSTraceHeader`**Systemattribut — Das `AWSTraceHeader` ist ein Nachrichtensystemattribut, das von Amazon SQS reserviert wurde, um den X-Ray-Trace-Header mit Nachrichten in der Warteschlange zu übertragen. `AWSTraceHeader`steht auch dann zur Verfügung, wenn die automatische Instrumentierung über das X-Ray-SDK nicht aktiviert ist, z. B. wenn ein Tracing-SDK für eine neue Sprache erstellt wird. Wenn beide Header-Instrumentierungen festgelegt sind, überschreibt das Nachrichtensystemattribut den HTTP-Nachverfolgungs-Header.

Bei der Ausführung auf Amazon EC2 unterstützt Amazon SQS die Verarbeitung jeweils einer Nachricht. Dies gilt, wenn sie auf einem lokalen Host ausgeführt werden und wenn Container-Services wie AWS Fargate Amazon ECS oder AWS App Mesh verwendet werden.

Der Trace-Header ist sowohl von der Amazon SQS SQS-Nachrichtengröße als auch von den Nachrichtenattributen ausgenommen. Durch die Aktivierung von X-Ray Tracing werden Ihre Amazon SQS SQS-Kontingente nicht überschritten. Weitere Informationen zu AWS Kontingenten finden Sie unter Amazon SQS SQS-Kontingente.

Senden des HTTP-Nachverfolgungs-Headers

Absenderkomponenten in Amazon SQS können den Trace-Header automatisch über den `SendMessageBatchSendMessage`OR-Aufruf senden. Wenn AWS SDK-Clients instrumentiert sind, können sie automatisch in allen vom X-Ray SDK unterstützten Sprachen nachverfolgt werden.

Verfolgte Ressourcen AWS-Services und Ressourcen, auf die Sie innerhalb dieser Services zugreifen (z. B. ein Amazon S3 S3-Bucket oder eine Amazon SQS SQS-Warteschlange), werden als Downstream-Knoten auf der Trace-Map in der X-Ray-Konsole angezeigt.

Informationen zum Verfolgen von AWS SDK-Aufrufen in Ihrer bevorzugten Sprache finden Sie in den folgenden Themen in der unterstützten Version: SDKs

- Go – Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for Go
- Java — Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for Java
- Node.js – Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK für Node.js
- Python – Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK für Python
- Ruby – Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for Ruby
- .NET – Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for .NET

Abrufen des Nachverfolgungs-Headers und Wiederherstellen des Nachverfolgungskontexts

Wenn Sie einen Lambda-Downstream-Consumer verwenden, erfolgt die Übertragung des Trace-Kontextes automatisch. Um die Kontextweiterleitung mit anderen Amazon SQS SQS-Verbrauchern fortzusetzen, müssen Sie die Übergabe an die Empfängerkomponente manuell instrumentieren.

Es gibt drei Hauptschritte für die Wiederherstellung des Nachverfolgungskontexts:

- Empfangen Sie die Nachricht aus der Warteschlange für das `AWSTraceHeader`-Attribut, indem Sie die `ReceiveMessage`-API aufrufen.
- Rufen Sie den Nachverfolgungs-Header aus dem Attribut ab.
- Stellen Sie die Nachverfolgungs-ID aus dem Header wieder her. Optional können Sie dem Segment weitere Metriken hinzufügen.

Im Folgenden finden Sie eine Beispielimplementierung, die mit dem X-Ray SDK for Java geschrieben wurde.

Example Abrufen des Nachverfolgungs-Headers und Wiederherstellen des Nachverfolgungskontexts

```
// 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 und AWS X-Ray

AWS X-Ray lässt sich in Amazon S3 integrieren, um Upstream-Anfragen zur Aktualisierung der S3-Buckets Ihrer Anwendung zu verfolgen. Wenn ein Service Anfragen mithilfe des X-Ray-SDK verfolgt, kann Amazon S3 die Tracing-Header an Downstream-Event-Abonnenten wie AWS Lambda Amazon SQS und Amazon SNS senden. X-Ray aktiviert Trace-Nachrichten für Amazon-S3-Ereignisbenachrichtigungen.

Sie können die X-Ray-Trace-Karte verwenden, um die Verbindungen zwischen Amazon S3 und anderen Diensten, die Ihre Anwendung nutzt, anzuzeigen. Sie können mithilfe der Konsole auch Metriken, wie durchschnittliche Latenz- und Ausfallraten, anzuzeigen. Weitere Informationen zur X-Ray-Konsole finden Sie unterVerwenden Sie die X-Ray-Konsole.

Amazon S3 unterstützt die standardmäßige HTTP-Header-Instrumentierung. Das X-Ray-SDK füllt den Trace-Header automatisch als HTTP-Header auf, wenn Sie Amazon S3 über das AWS SDK aufrufen. Der Standard-Trace-Header wird von `X-Amzn-Trace-Id` übernommen. Weitere Informationen zu Tracing-Headern finden Sie Ablaufverfolgungs-Header auf der Konzeptseite. Amazon S3 Trace Context Propagation unterstützt die folgenden Abonnenten: Lambda, SQS und SNS. Da SQS und SNS selbst keine Segmentdaten ausgeben, werden sie nicht in Ihrer Trace-

oder Trace-Map angezeigt, wenn sie von S3 ausgelöst werden, obwohl sie den Tracing-Header an nachgeschaltete Dienste weitergeben.

Amazon S3 S3-Ereignisbenachrichtigungen konfigurieren

Mit der Amazon S3 S3-Benachrichtigungsfunktion erhalten Sie Benachrichtigungen, wenn bestimmte Ereignisse in Ihrem Bucket eintreten. Diese Benachrichtigungen können dann an die folgenden Ziele innerhalb Ihrer Anwendung weitergegeben werden:

- Amazon-Simple-Notification-Service (Amazon-SNS)
- Amazon-Simple-Queue-Service (Amazon SQS)
- AWS Lambda

Eine Liste der unterstützten Ereignisse finden Sie unter Unterstützte Ereignistypen im Amazon S3 S3-Entwicklerhandbuch.

Amazon SNS und Amazon SQS

Um Benachrichtigungen zu einem SNS-Thema oder einer SQS-Warteschlange zu veröffentlichen, müssen Sie zunächst Amazon S3 S3-Berechtigungen erteilen. Um diese Berechtigungen zu gewähren, fügen Sie dem Ziel-SNS-Thema oder der SQS-Warteschlange eine AWS Identity and Access Management (IAM-) Richtlinie hinzu. Weitere Informationen zu den erforderlichen IAM-Richtlinien finden Sie unter Erteilen von Berechtigungen zum Veröffentlichen von Nachrichten in einem SNS-Thema oder einer SQS-Warteschlange.

Informationen zur Integration von SNS und SQS mit X-Ray finden Sie unter und. Amazon SNS und AWS X-Ray Amazon SQS und AWS X-Ray

AWS Lambda

Wenn Sie die Amazon S3-Konsole verwenden, um Ereignisbenachrichtigungen in einem S3-Bucket für eine Lambda-Funktion zu konfigurieren, richtet die Konsole die erforderlichen Berechtigungen für die Lambda-Funktion ein, sodass Amazon S3 berechtigt ist, die Funktion aus dem Bucket aufzurufen. Weitere Informationen finden Sie unter Wie aktiviere und konfiguriere ich Ereignisbenachrichtigungen für einen S3-Bucket? im Amazon Simple Storage Service Console-Benutzerhandbuch.

Sie können Amazon S3 S3-Berechtigungen auch erteilen AWS Lambda , um Ihre Lambda-Funktion aufzurufen. Weitere Informationen finden Sie unter Tutorial: Using AWS Lambda with Amazon S3 im AWS Lambda Developer Guide.

Weitere Informationen zur Integration von Lambda mit X-Ray finden Sie unter Instrumentieren von Java-Code in AWS Lambda.

AWS Distribution für und OpenTelemetry AWS X-Ray

Verwenden Sie AWS Distro for OpenTelemetry (ADOT), um Metriken und Traces zu sammeln AWS X-Ray und an andere Überwachungslösungen wie Amazon CloudWatch, Amazon Service und Amazon Managed OpenSearch Service for Prometheus zu senden.

AWS Distro für OpenTelemetry

The AWS Distro for OpenTelemetry (ADOT) ist eine AWS Distribution, die auf dem Projekt der Cloud Native Computing Foundation (CNCF) basiert. OpenTelemetry OpenTelemetry bietet einen einzigen Satz von Open-Source-Dateien APIs, Bibliotheken und Agenten zur Erfassung verteilter Traces und Metriken. Dieses Toolkit ist eine Distribution von OpenTelemetry Upstream-Komponenten SDKs, einschließlich Agenten für automatische Instrumentierung und Kollektoren, die von getestet, optimiert, gesichert und unterstützt werden. AWS

Mit ADOT können Techniker ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Überwachungslösungen senden, darunter Amazon CloudWatch AWS X-Ray, Amazon Service und Amazon Managed OpenSearch Service for Prometheus.

ADOT ist in eine wachsende Anzahl von integriert, AWS-Services um das Senden von Traces und Metriken an Monitoring-Lösungen wie X-Ray zu vereinfachen. Einige Beispiele für Dienste, die in ADOT integriert sind, sind:

- AWS Lambda— AWS verwaltete Lambda-Schichten für ADOT bieten eine plug-and-play Benutzererfahrung, indem sie eine Lambda-Funktion automatisch instrumentieren und OpenTelemetry zusammen mit einer out-of-the-box Konfiguration für AWS Lambda und X-Ray in einer einfach einzurichtenden Ebene zusammenfassen. Benutzer können ihre Lambda-Funktion aktivieren und deaktivieren OpenTelemetry , ohne den Code zu ändern. Weitere Informationen finden Sie unter AWS Distro for Lambda OpenTelemetry
- Amazon Elastic Container Service (ECS) — Erfassen Sie mit AWS Distro for OpenTelemetry Collector Metriken und Traces aus Amazon ECS-Anwendungen, um sie an X-Ray und andere Überwachungslösungen zu senden. Weitere Informationen finden Sie unter Sammeln von Anwendungs-Trace-Daten im Amazon ECS-Entwicklerhandbuch.
- AWS App Runner — App Runner unterstützt das Senden von Traces an X-Ray mithilfe der AWS Distro for OpenTelemetry (ADOT). Verwenden Sie ADOT SDKs , um Trace-Daten für Ihre

containerisierten Anwendungen zu sammeln, und verwenden Sie X-Ray, um Ihre instrumentierte Anwendung zu analysieren und Einblicke in sie zu gewinnen. Weitere Informationen finden Sie unter AWS App Runner und X-Ray.

Weitere Informationen zur AWS Distribution für OpenTelemetry, einschließlich der Integration mit Zusatzsoftware AWS-Services, finden Sie in der Dokumentation zur AWS OpenTelemetry Distribution.

Weitere Informationen zur Instrumentierung Ihrer Anwendung mit AWS Distro for OpenTelemetry und X-Ray finden Sie unter Instrumentieren Ihrer Anwendung mit Distro for. AWS OpenTelemetry

Verfolgen von Konfigurationsänderungen bei der X-Ray-Verschlüsselung mit AWS Config

AWS X-Ray integriert sich in AWS Config , um Konfigurationsänderungen aufzuzeichnen, die an Ihren X-Ray-Verschlüsselungsressourcen vorgenommen wurden. Sie können AWS Config damit Ressourcen für die X-Ray-Verschlüsselung inventarisieren, den X-Ray-Konfigurationsverlauf überprüfen und Benachrichtigungen auf Grundlage von Ressourcenänderungen versenden.

AWS Config unterstützt die Protokollierung der folgenden Änderungen der X-Ray-Verschlüsselungsressourcen als Ereignisse:

- Konfigurationsänderungen — Ändern oder Hinzufügen eines Verschlüsselungsschlüssels oder Zurücksetzen auf die Standard-X-Ray-Verschlüsselungseinstellung.

Verwenden Sie die folgenden Anweisungen, um zu erfahren, wie Sie eine grundlegende Verbindung zwischen X-Ray und herstellen AWS Config.

Einen Lambda-Funktionstrigger erstellen

Sie benötigen den ARN einer benutzerdefinierten AWS Lambda Funktion, bevor Sie eine benutzerdefinierte AWS Config Regel generieren können. Befolgen Sie diese Anweisungen zum Erstellen einer einfachen Funktion mit Node.js, die basierend auf dem Zustand der `XrayEncryptionConfig`-Ressource als Wert an AWS Config zurückgibt, ob sie konform oder nicht konform ist.

Um eine Lambda-Funktion mit einem AWS::Xray EncryptionConfig Change-Trigger zu erstellen

1. Öffnen Sie die Lambda-Konsole. Wählen Sie Funktion erstellen aus.
2. Wählen Sie Blueprints aus, und filtern Sie dann die Blueprints-Bibliothek nach dem Blueprint. config-rule-change-triggered Klicken Sie auf den Link im Namen der Vorlage oder wählen Sie Configure (Konfigurieren), um fortzufahren.
3. Definieren Sie die folgenden Felder zum Konfigurieren der Vorlage:
   - Geben Sie im Feld Name einen Namen ein.
   - Für die Option Role (Rolle) wählen Sie Create new role from template(s) (Neue Rolle aus Vorlage[n] erstellen) aus.
   - Bei Role Name geben Sie einen Namen ein.
   - Wählen Sie für Policy Templates (Richtlinienvorlagen) die Option AWS Config Rules permissions (-Regelberechtigungen) aus.
4. Wählen Sie Funktion erstellen, um Ihre Funktion zu erstellen und in der Konsole anzuzeigen. AWS Lambda
5. Bearbeiten Sie Ihren Funktionscode, indem Sie `AWS::EC2::Instance` durch `AWS::XrayEncryptionConfig` ersetzen. Sie können auch das Beschreibungsfeld aktualisieren, damit die Änderung wirksam wird.

   Standardcode

```
    if (configurationItem.resourceType !== 'AWS::EC2::Instance') {
        return 'NOT_APPLICABLE';
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
        return 'NON_COMPLIANT';
```

   Aktualisierter Code

```
    if (configurationItem.resourceType !== 'AWS::XRay::EncryptionConfig') {
        return 'NOT_APPLICABLE';
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
```

```
        return 'NON_COMPLIANT';
```

6. Fügen Sie Ihrer Ausführungsrolle in IAM Folgendes hinzu, um auf X-Ray zugreifen zu können. Diese Berechtigungen ermöglichen den Nur-Lese-Zugriff auf Ihre X-Ray-Ressourcen. Wenn kein Zugriff auf die entsprechenden Ressourcen gewährt wird, wird bei der Auswertung der mit der Regel verknüpften AWS Config Lambda-Funktion eine Meldung außerhalb des Gültigkeitsbereichs angezeigt.

```
{
    "Sid": "Stmt1529350291539",
    "Action": [
        "xray:GetEncryptionConfig"
    ],
    "Effect": "Allow",
    "Resource": "*"
 }
```

Eine benutzerdefinierte AWS Config Regel für X-Ray erstellen

Wenn die Lambda-Funktion erstellt wurde, notieren Sie sich den ARN der Funktion und rufen Sie die AWS Config Konsole auf, um Ihre benutzerdefinierte Regel zu erstellen.

Um eine AWS Config Regel für X-Ray zu erstellen

1. Öffnen Sie die Seite Rules (Regeln) der AWS Config -Konsole.
2. Klicken Sie auf Add Rule (Regel hinzufügen) und danach auf Add custom rule (Benutzerdefinierte Regel hinzufügen).
3. Fügen Sie in AWS Lambda Function ARN den ARN ein, der der Lambda-Funktion zugeordnet ist, die Sie verwenden möchten.
4. Wählen Sie, welche Art von Auslöser festgelegt werden soll:

   - Konfigurationsänderungen — AWS Config löst die Auswertung aus, wenn sich die Konfiguration einer Ressource ändert, die dem Geltungsbereich der Regel entspricht. Die Evaluierung wird ausgeführt, nachdem eine Benachrichtigung über eine Änderung des Konfigurationselements AWS Config gesendet wurde.
   - Periodisch — AWS Config Führt Evaluierungen für die Regel mit einer von Ihnen gewählten Häufigkeit aus (z. B. alle 24 Stunden).

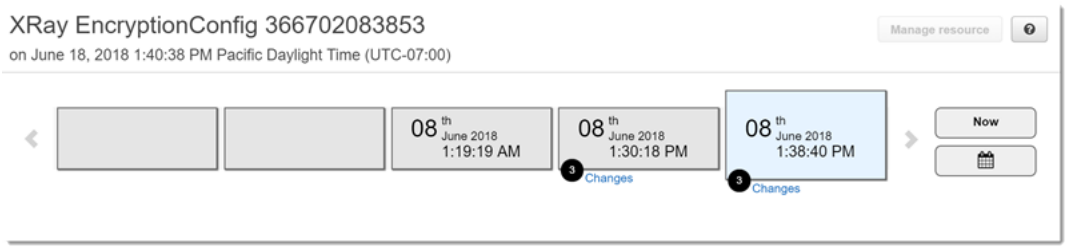
5. Wählen Sie für Resource type (Ressourcentyp) die Option EncryptionConfigin der Röntgenabteilung.
6. Wählen Sie Save (Speichern) aus.

Die AWS Config Konsole beginnt sofort mit der Bewertung der Regelkonformität. Die Auswertung kann mehrere Minuten in Anspruch nehmen.

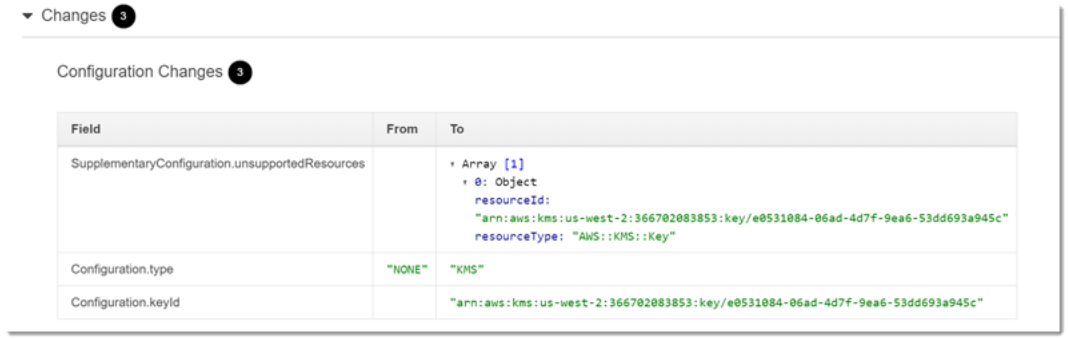
Da diese Regel nun konform ist, AWS Config kann mit der Erstellung eines Auditverlaufs begonnen werden. AWS Config zeichnet Ressourcenänderungen in Form einer Zeitleiste auf. AWS Config Generiert für jede Änderung in der Zeitleiste der Ereignisse eine Tabelle im Von/bis-Format, aus der hervorgeht, was sich an der JSON-Darstellung des Verschlüsselungsschlüssels geändert hat. Die beiden damit verbundenen Feldänderungen EncryptionConfig sind `Configuration.type` und. `Configuration.keyID`

Beispielergebnisse

Im Folgenden finden Sie ein Beispiel für eine AWS Config Zeitleiste, in der Änderungen angezeigt werden, die an bestimmten Tagen und zu bestimmten Zeiten vorgenommen wurden.



Es folgt ein Beispiel für einen AWS Config Änderungseintrag. Das Format "Von/Zu" veranschaulicht, was sich geändert hat. Dieses Beispiel zeigt, dass die Standardeinstellungen für die X-Ray-Verschlüsselung auf einen definierten Verschlüsselungsschlüssel geändert wurden.



Amazon-SNS-Benachrichtigungen

Um über Konfigurationsänderungen informiert zu werden, stellen Sie AWS Config die Veröffentlichung von Amazon SNS SNS-Benachrichtigungen ein. Weitere Informationen finden Sie unter Überwachen von AWS Config Ressourcenänderungen per E-Mail.

AWS AppSync und AWS X-Ray

Sie können Anfragen für AWS AppSync aktivieren und verfolgen. Weitere Informationen finden Sie unter Tracing with AWS X-Ray für Anweisungen.

Wenn X-Ray Tracing für eine AWS AppSync API aktiviert ist, wird in Ihrem Konto automatisch eine mit dem Dienst verknüpfte AWS Identity and Access Management Zugriffsverwaltungsrolle mit den entsprechenden Berechtigungen erstellt. Dies ermöglicht es AWS AppSync , Spuren auf sichere Weise an X-Ray zu senden.

Unterstützung für aktives Tracing von Amazon API Gateway für AWS X-Ray

Sie können X-Ray verwenden, um Benutzeranfragen zu verfolgen und zu analysieren, während sie über Ihr Amazon API Gateway APIs zu den zugrunde liegenden Diensten weitergeleitet werden. API Gateway unterstützt X-Ray Tracing für alle API Gateway Gateway-Endpunkttypen: Regional, Edge-optimiert und privat. Sie können X-Ray mit Amazon API Gateway überall dort verwenden AWS-Regionen , wo X-Ray verfügbar ist. Weitere Informationen finden Sie unter Trace API Gateway API Execution with AWS X-Ray im Amazon API Gateway Developer Guide.

> **Note**
>
> X-Ray unterstützt nur Tracing für REST APIs über API Gateway.

Amazon API Gateway bietet aktive Ablaufverfolgungsunterstützung für AWS X-Ray. Aktivieren Sie aktives Tracing auf Ihren API-Stufen, um eingehende Anfragen zu testen und Traces an X-Ray zu senden.

So aktivieren Sie die aktive Ablaufverfolgung auf einer API-Stufe

1. Öffnen Sie die API Gateway Gateway-Konsole unter https://console.aws.amazon.com/apigateway/.
2. Wählen Sie eine API aus.
3. Wählen Sie eine Stufe aus.
4. Wählen Sie auf der Registerkarte Logs/Tracing die Option X-Ray Tracing aktivieren und dann Änderungen speichern aus.
5. Wählen Sie im Navigationsbereich links Resources (Ressourcen) aus.
6. Um die API mit den neuen Einstellungen erneut bereitzustellen, wählen Sie das Drop-down-Menü „Aktionen“ und dann „API bereitstellen“.

API Gateway verwendet Sampling-Regeln, die Sie in der X-Ray-Konsole definieren, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen. Sie können Regeln erstellen, die nur für Anfragen gelten APIs, oder die nur für Anfragen gelten, die bestimmte Header enthalten. API Gateway zeichnet Header in Attributen des Segments zusammen mit Details zur Phase und Anfrage auf. Weitere Informationen finden Sie unter Konfigurieren von -Samplingregeln.

> **Note**
>
> Bei der Rückverfolgung von REST APIs mit der API Gateway-HTTP-Integration wird der Dienstname jedes Segments auf den Anforderungs-URL-Pfad von API Gateway zu Ihrem HTTP-Integrationsendpunkt gesetzt, was zu einem Dienstknoten auf der X-Ray-Trace-Map für jeden eindeutigen URL-Pfad führt. Eine große Anzahl von URL-Pfaden kann dazu führen, dass die Trace-Map die Grenze von 10.000 Knoten überschreitet, was zu einem Fehler führt. Um die Anzahl der von API Gateway erstellten Dienstknoten zu minimieren, sollten Sie erwägen, Parameter innerhalb der URL-Abfragezeichenfolge oder im Anforderungstext per POST zu übergeben. Bei beiden Ansätzen wird sichergestellt, dass Parameter nicht Teil des URL-Pfads sind, was zu weniger eindeutigen URL-Pfaden und Dienstknoten führen kann.

Für alle eingehenden Anfragen fügt API Gateway eingehenden HTTP-Anfragen, die noch keinen haben, einen Tracing-Header hinzu.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Röntgen-Trace-ID-Format

Ein X-Ray `trace_id` besteht aus drei Zahlen, die durch Bindestriche getrennt sind. Beispiel, `1-58406520-a006649127e371903a2de979`. Dies umfasst:

- Die Versionsnummer, die ist. 1
- Die Uhrzeit der ursprünglichen Anfrage in Unix-Epochenzeit unter Verwendung von 8 Hexadezimalziffern.

  Zum Beispiel ist 10:00 Uhr am 1. Dezember 2016 PST in Epochenzeit `1480615200` Sekunden oder `58406520` in Hexadezimalziffern.
- Eine weltweit eindeutige 96-Bit-ID für den Trace mit 24 Hexadezimalziffern.

Wenn die aktive Ablaufverfolgung deaktiviert ist, zeichnet die Stufe dennoch ein Segment auf, wenn die Anforderung von einem Service stammt, der die Anforderung geprüft und eine Nachverfolgung gestartet hat. Beispielsweise kann eine instrumentierte Webanwendung eine API Gateway mit einem HTTP-Client aufrufen. Wenn Sie einen HTTP-Client mit dem X-Ray SDK instrumentieren, fügt es der ausgehenden Anfrage einen Tracing-Header hinzu, der die Sampling-Entscheidung enthält. API Gateway liest den Tracing-Header und erstellt ein Segment für gesampelte Anfragen.

Wenn Sie API Gateway verwenden, um ein Java-SDK für Ihre API zu generieren, können Sie den SDK-Client instrumentieren, indem Sie einen Request-Handler mit dem Client Builder hinzufügen, genauso wie Sie einen AWS SDK-Client manuell instrumentieren würden. Detaillierte Anweisungen finden Sie unter Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for Java.

Amazon EC2 und AWS App Mesh

AWS X-Ray lässt sich in Envoy-Proxys für Microservices integrieren, um sie AWS App Meshzu verwalten. App Mesh bietet eine Version von Envoy, die Sie so konfigurieren können, dass Trace-Daten an den X-Ray-Daemon gesendet werden, der in einem Container derselben Aufgabe oder desselben Pods ausgeführt wird. X-Ray unterstützt die Ablaufverfolgung mit den folgenden App Mesh Mesh-kompatiblen Diensten:

- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Amazon Elastic Compute Cloud (Amazon EC2)

Verwenden Sie die folgenden Anweisungen, um zu erfahren, wie Sie das X-Ray-Tracing durch App Mesh aktivieren.

Um den Envoy-Proxy so zu konfigurieren, dass er Daten an X-Ray sendet, legen Sie die `ENABLE_ENVOY_XRAY_TRACING` Umgebungsvariable in seiner Container-Definition fest.

> **Note**
>
> Die App Mesh Mesh-Version von Envoy sendet derzeit keine Traces, die auf konfigurierten Sampling-Regeln basieren. Stattdessen verwendet sie eine feste Samplingrate von 5% für

Envoy-Version 1.16.3 oder neuer oder eine 50-prozentige Samplingrate für Envoy-Versionen vor 1.16.3.

Example Envoy-Container-Definition für 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

Weitere Informationen zu den verfügbaren Envoy-Regionsadressen finden Sie im Envoy-Bild im AWS App Mesh Benutzerhandbuch.

Einzelheiten zur Ausführung des X-Ray-Daemons in einem Container finden Sie unterDen X-Ray-Daemon auf Amazon ECS ausführen. Für eine Beispielanwendung, die ein Service Mesh, einen Microservice, einen Envoy-Proxy und einen X-Ray-Daemon umfasst, stellen Sie das `colorapp` Beispiel im App Mesh Examples Repository bereit. GitHub

Weitere Informationen

- Erste Schritte mit AWS App Mesh
- Erste Schritte mit AWS App Mesh Amazon ECS

AWS App Runner und X-Ray

AWS App Runner bietet eine AWS-Service schnelle, einfache und kostengünstige Möglichkeit, Quellcode oder ein Container-Image direkt in eine skalierbare und sichere Webanwendung in der zu implementieren. AWS Cloud Sie müssen sich nicht mit neuen Technologien vertraut machen, entscheiden, welchen Rechendienst Sie verwenden möchten, oder wissen, wie AWS Ressourcen bereitgestellt und konfiguriert werden. Weitere Informationen finden Sie unter Was ist AWS App Runner.

AWS App Runner sendet Traces an X-Ray, indem es in die AWS Distro for OpenTelemetry (ADOT) integriert wird. Verwenden Sie ADOT SDKs , um Trace-Daten für Ihre containerisierten Anwendungen zu sammeln, und verwenden Sie X-Ray, um Ihre instrumentierte Anwendung zu analysieren und Einblicke in sie zu gewinnen. Weitere Informationen finden Sie unter Tracing für Ihre App Runner-Anwendung mit X-Ray.

Protokollierung von X-Ray-API-Aufrufen mit AWS CloudTrail

AWS X-Ray ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt AWS-Service. CloudTrail erfasst alle API-Aufrufe für X-Ray als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der X-Ray-Konsole und Code-Aufrufe der X-Ray-API-Operationen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an X-Ray gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.

- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit dem CloudTrail Ereignisverlauf. Für die Anzeige des Ereignisverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder CloudTrailLake-Event-Datenspeicher.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS-Managementkonsole sind regionsübergreifend. Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter Erstellen eines Trails für Ihr AWS-Konto und Erstellen eines Trails für eine Organisation im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise. Informationen zu Amazon-S3-Preisen finden Sie unter Amazon S3 – Preise.

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format. ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von erweiterten Ereignisselektoren auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter Arbeiten mit AWS CloudTrail Lake im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die Preisoption aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise.

Themen

- X-Ray-Management-Ereignisse in CloudTrail
- Röntgendatenereignisse in CloudTrail
- Beispiele für X-Ray-Ereignisse

X-Ray-Management-Ereignisse in CloudTrail

AWS X-Ray lässt sich integrieren AWS CloudTrail , um API-Aktionen aufzuzeichnen, die von einem Benutzer, einer Rolle oder einem AWS-Service in X-Ray ausgeführt wurden. Sie können CloudTrail damit X-Ray-API-Anfragen in Echtzeit überwachen und Protokolle in Amazon S3, Amazon CloudWatch Logs und Amazon CloudWatch Events speichern. X-Ray unterstützt die Protokollierung der folgenden Aktionen als Ereignisse in CloudTrail Protokolldateien:

Unterstützte API-Aktionen

- PutEncryptionConfig
- GetEncryptionConfig
- CreateGroup
- UpdateGroup
- DeleteGroup
- GetGroup
- GetGroups
- GetInsight
- GetInsightEvents
- GetInsightImpactGraph
- GetInsightSummaries

- GetSamplingStatisticSummaries

Röntgendatenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die an oder in einer Ressource ausgeführt werden (z. B. PutTraceSegments, das Segmentdokumente auf X-Ray hochlädt).

Sie werden auch als Vorgänge auf Datenebene bezeichnet. Datenereignisse sind oft Aktivitäten mit hohem Volume. Protokolliert standardmäßig CloudTrail keine Datenereignisse. Der CloudTrail Ereignisverlauf zeichnet keine Datenereignisse auf.

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preisgestaltung.

Sie können Datenereignisse für die X-Ray-Ressourcentypen mithilfe der CloudTrail Konsole oder CloudTrail API-Operationen protokollieren. AWS CLI Weitere Informationen zum Protokollieren von Datenereignissen finden Sie unter Protokollieren von Datenereignissen mit dem AWS-Managementkonsole und Protokollieren von Datenereignissen mit dem AWS Command Line Interface im AWS CloudTrail -Benutzerhandbuch.

In der folgenden Tabelle sind die X-Ray-Ressourcentypen aufgeführt, für die Sie Datenereignisse protokollieren können. In der Spalte Datenereignistyp (Konsole) wird der Wert angezeigt, den Sie in der Liste Datenereignistyp auf der CloudTrail Konsole auswählen können. In der Wertspalte resources.type wird der `resources.type` Wert angezeigt, den Sie bei der Konfiguration erweiterter Event-Selektoren mithilfe von oder angeben würden. AWS CLI CloudTrail APIs In der CloudTrail Spalte APIs Protokollierte Daten werden die API-Aufrufe angezeigt, die CloudTrail für den Ressourcentyp protokolliert wurden.

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail
Röntgenspur	`AWS::XRay::Trace`	• PutTraceSegments • GetTraceSummaries • GetTraceGraph • GetServiceGraph • BatchGetTraces

Typ des Datenereignisses (Konsole)	resources.type-Wert	Daten, die APIs protokolliert wurden CloudTrail
		• GetTimeSeriesServiceStatistics • PutTelemetryRecords • GetSamplingTargets

Sie können erweiterte Event-Selektoren so konfigurieren, dass sie nach den `readOnly` Feldern `eventName` und filtern, sodass nur die Ereignisse protokolliert werden, die für Sie wichtig sind. Sie können jedoch keine Ereignisse auswählen, indem Sie den `resources.ARN` Feldselektor hinzufügen, da dies bei Röntgenspuren nicht der Fall ist. ARNs Weitere Informationen zu diesen Feldern finden Sie unter AdvancedFieldSelector in der API-Referenz zu AWS CloudTrail Im Folgenden finden Sie ein Beispiel für die Ausführung des `put-event-selectors` AWS CLI Befehls zum Protokollieren von Datenereignissen in einem CloudTrail Trail. Sie müssen den Befehl in der Region ausführen oder die Region angeben, in der der Trail erstellt wurde. Andernfalls gibt der Vorgang eine `InvalidHomeRegionException` Ausnahme zurück.

```
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"
      }
   ]
}'
```

Beispiele für X-Ray-Ereignisse

Beispiel für ein Management-Event, **`GetEncryptionConfig`**

Das Folgende ist ein Beispiel für das X-Ray GetEncryptionConfig Eintrag einloggen 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"
}
```

Beispiel für ein Datenereignis, **`PutTraceSegments`**

Das Folgende ist ein Beispiel für das X-Ray PutTraceSegments Eintrag in das Datenereignisprotokoll 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"
  }
}
```

CloudWatch Integration mit X-Ray

AWS X-Ray lässt sich in CloudWatch Application Signals, CloudWatch RUM und CloudWatch Synthetics integrieren, um die Überwachung des Zustands Ihrer Anwendungen zu vereinfachen. Ermöglichen Sie es Ihrer Anwendung für Application Signals, den Betriebsstatus Ihrer Dienste, Clientseiten, Synthetics Canaries und Serviceabhängigkeiten zu überwachen und Fehler zu beheben.

Durch die Korrelation von CloudWatch Metriken, Protokollen und X-Ray-Traces bietet die X-Ray-Trace-Map einen end-to-end Überblick über Ihre Services, sodass Sie Leistungsengpässe schnell erkennen und betroffene Benutzer identifizieren können.

Mit CloudWatch RUM können Sie eine echte Benutzerüberwachung durchführen, um clientseitige Daten über die Leistung Ihrer Webanwendung aus tatsächlichen Benutzersitzungen nahezu in Echtzeit zu sammeln und anzuzeigen. Mit AWS X-Ray und CloudWatch RUM können Sie den Anforderungspfad analysieren und debuggen, angefangen bei den Endbenutzern Ihrer Anwendung bis hin zu nachgelagerten Managed Services. AWS Auf diese Weise können Sie Latenztrends und Fehler identifizieren, die sich auf Ihre Endbenutzer auswirken.

Themen

- CloudWatch RUM und AWS X-Ray
- Debuggen von CloudWatch synthetischen Kanarienvögeln mit X-Ray

CloudWatch RUM und AWS X-Ray

Mit Amazon CloudWatch RUM können Sie eine echte Benutzerüberwachung durchführen, um clientseitige Daten über die Leistung Ihrer Webanwendung aus tatsächlichen Benutzersitzungen nahezu in Echtzeit zu sammeln und anzuzeigen. Mit AWS X-Ray und CloudWatch RUM können Sie den Anforderungspfad analysieren und debuggen, angefangen bei den Endbenutzern Ihrer Anwendung bis hin zu nachgelagerten Managed Services. AWS Auf diese Weise können Sie Latenztrends und Fehler identifizieren, die sich auf Ihre Endbenutzer auswirken.

Nachdem Sie die Röntgenverfolgung von Benutzersitzungen aktiviert haben, fügt CloudWatch RUM den zulässigen HTTP-Anfragen einen X-Ray-Trace-Header hinzu und zeichnet ein X-Ray-Segment für zulässige HTTP-Anfragen auf. Sie können dann Traces und Segmente aus diesen Benutzersitzungen im X-Ray und in den CloudWatch Konsolen sehen, einschließlich der X-Ray-Trace-Map.

> **Note**
>
> CloudWatch RUM lässt sich nicht in die X-Ray-Probenahmeregeln integrieren. Wählen Sie stattdessen einen Prozentsatz für die Probenahme aus, wenn Sie Ihre Anwendung für die Verwendung von CloudWatch RUM einrichten. Bei Traces, die von CloudWatch RUM gesendet werden, können zusätzliche Kosten anfallen. Weitere Informationen finden Sie unter AWS X-Ray Preise.

Standardmäßig sind von CloudWatch RUM gesendete clientseitige Traces nicht mit serverseitigen Traces verbunden. Um clientseitige Traces mit serverseitigen Traces zu verbinden, konfigurieren Sie den CloudWatch RUM-Webclient so, dass er diesen HTTP-Anfragen einen X-Ray-Trace-Header hinzufügt.

> **Warning**
>
> Wenn Sie den CloudWatch RUM-Webclient so konfigurieren, dass er HTTP-Anfragen einen X-Ray-Trace-Header hinzufügt, kann dies dazu führen, dass Cross-Origin Resource Sharing (CORS) fehlschlägt. Um dies zu vermeiden, fügen Sie den `X-Amzn-Trace-Id` HTTP-Header der Liste der zulässigen Header in der CORS-Konfiguration Ihres Downstream-Dienstes hinzu. Wenn Sie API Gateway als Downstream verwenden, finden Sie weitere Informationen unter CORS für eine REST-API-Ressource aktivieren. Wir empfehlen dringend, Ihre Anwendung zu testen, bevor Sie einen clientseitigen X-Ray-Ablaufverfolgungs-

Header in einer Produktionsumgebung hinzufügen. Weitere Informationen finden Sie in der CloudWatch RUM-Webclient-Dokumentation.

Weitere Informationen zur Überwachung von echten Benutzern finden Sie unter CloudWatch RUM verwenden. CloudWatch Informationen zum Einrichten Ihrer Anwendung für die Verwendung von CloudWatch RUM, einschließlich der Protokollierung von Benutzersitzungen mit X-Ray, finden Sie unter Eine Anwendung für die Verwendung von CloudWatch RUM einrichten.

Debuggen von CloudWatch synthetischen Kanarienvögeln mit X-Ray

CloudWatch Synthetics ist ein vollständig verwalteter Service, mit dem Sie Ihre Endgeräte überwachen und Scripted Canaries APIs verwenden können, die 24 Stunden am Tag und einmal pro Minute ausgeführt werden.

Sie können Canary-Skripte anpassen, um nach Änderungen zu suchen in:

- Verfügbarkeit
- Latency
- Transaktionen
- Unterbrochene oder tote Links
- Step-by-step Erledigung von Aufgaben
- Fehler beim Laden der Seite
- Ladelatenzen für Komponenten der Benutzeroberfläche
- Komplexe Assistentenabläufe
- Checkout-Abläufe in Ihrer Anwendung

Canarys folgen denselben Routen, agieren und verhalten sich wie Ihre Kunden und überprüfen die Kundenerfahrung kontinuierlich.

Weitere Informationen zum Einrichten von Synthetics-Tests finden Sie unter Erstellen und Verwalten von Canarys mit Synthetics.

Die folgenden Beispiele zeigen häufige Anwendungsfälle für Debugging-Probleme, die von Ihren Synthetics-Canarys ausgelöst werden. Jedes Beispiel zeigt eine wichtige Strategie für das Debuggen mit der Trace-Map oder der X-Ray Analytics-Konsole.

Weitere Informationen zum Lesen und Interagieren mit der Trace-Map finden Sie unter Service-Map anzeigen.

Weitere Informationen zum Lesen und Interagieren mit der X-Ray Analytics-Konsole finden Sie unter Interaktion mit der AWS X-Ray Analytics-Konsole.

Themen

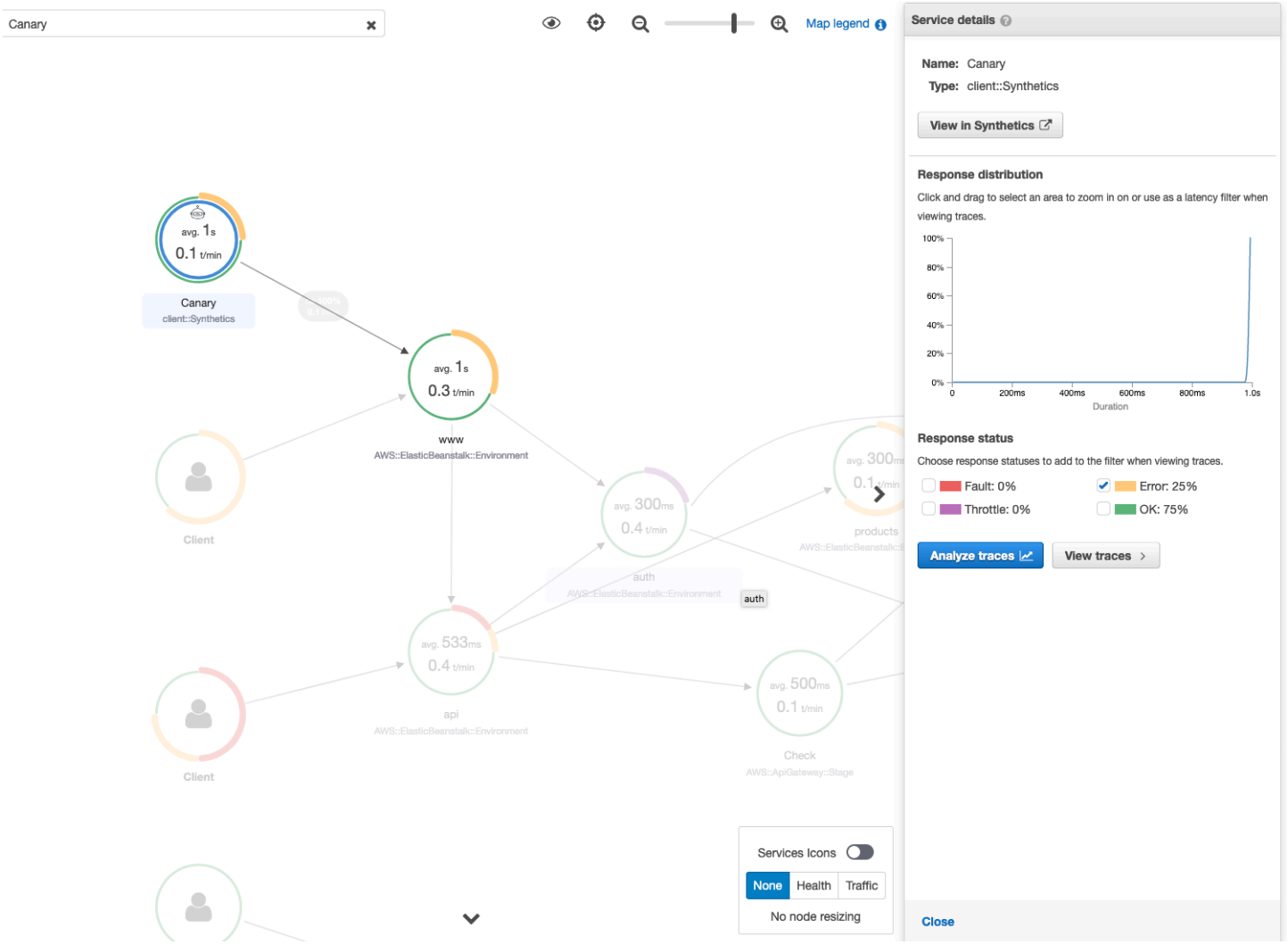
- In der Trace-Map werden Kanarienvögel angezeigt, bei denen häufiger Fehler gemeldet werden
- Verwenden Sie Trace-Detailkarten für einzelne Traces, um sich jede Anfrage im Detail anzusehen

- Die Ursache fortlaufender Fehler in Upstream- und Downstream-Services ermitteln
- Performance-Engpässe und Trends identifizieren
- Latenz und Fehler- oder Ausfallraten vor und nach Änderungen vergleichen
- Ermitteln Sie die erforderliche Kanarienabdeckung für alle APIs und URLs
- Gruppen für die Konzentration auf Synthetics-Tests verwenden

In der Trace-Map werden Kanarienvögel angezeigt, bei denen häufiger Fehler gemeldet werden

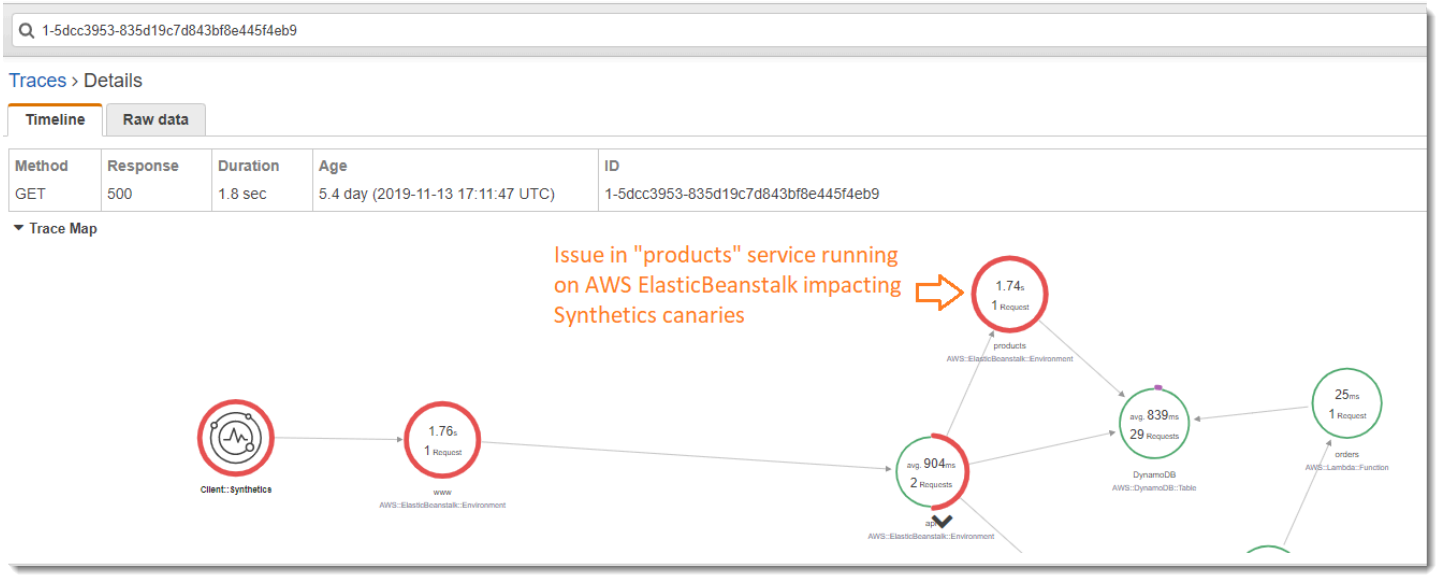
Um zu sehen, welche Kanaren in Ihrer X-Ray-Trace-Map häufiger Fehler, Störungen, Drosselungsraten oder langsame Reaktionszeiten aufweisen, können Sie die Canary-Clientknoten von Synthetics mithilfe des Filters hervorheben. `Client::Synthetic` Wenn Sie auf einen Knoten klicken, wird die Verteilung der Antwortzeit der gesamten Anfrage angezeigt. Wenn Sie auf eine Kante zwischen zwei Knoten klicken, werden Details zu den Anfragen angezeigt, die diese Verbindung verarbeitet haben. Sie können in Ihrer Trace-Map auch abgeleitete „entfernte“ Knoten für verwandte Downstream-Dienste anzeigen.

Wenn Sie auf den Synthetics-Knoten klicken, wird im Seitenbereich die Schaltfläche In Synthetics anzeigen angezeigt, über die Sie zur Synthetics-Konsole weitergeleitet werden, wo Sie die Canary-Details überprüfen können.



Verwenden Sie Trace-Detailkarten für einzelne Traces, um sich jede Anfrage im Detail anzusehen

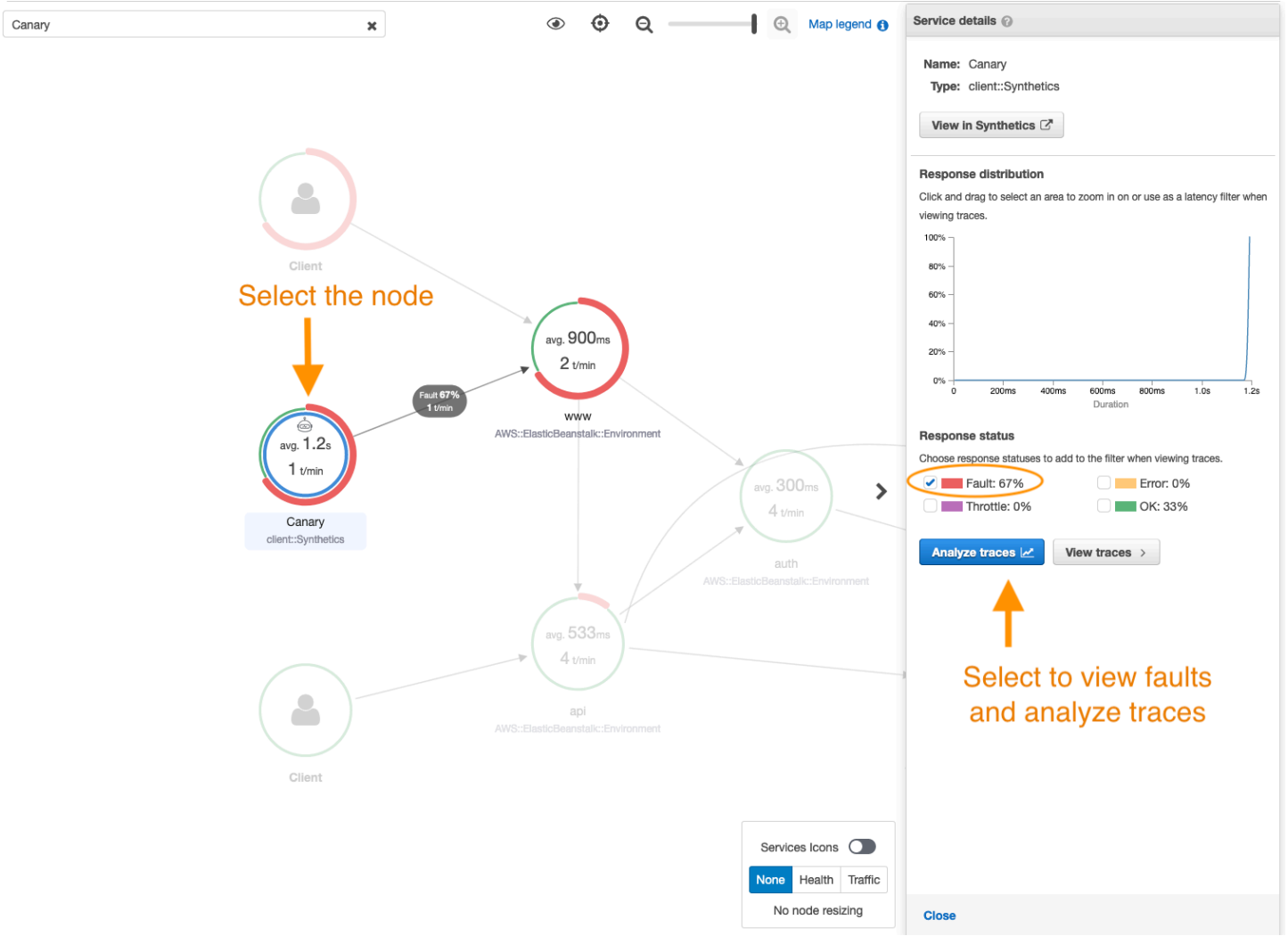
Um zu ermitteln, welcher Service zu der höchsten Latenz führt oder einen Fehler verursacht, rufen Sie die Trace-Details-Map auf, indem Sie die Trace in der Trace-Map auswählen. Die einzelnen Trace-Detailkarten zeigen den end-to-end Pfad einer einzelnen Anfrage an. Mit dieser Option können Sie die aufgerufenen Services verstehen und die Upstream- und Dowstream-Services visualisieren.

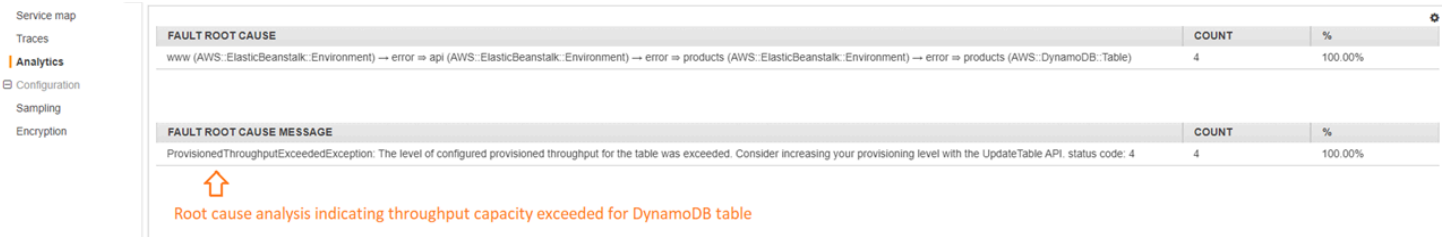


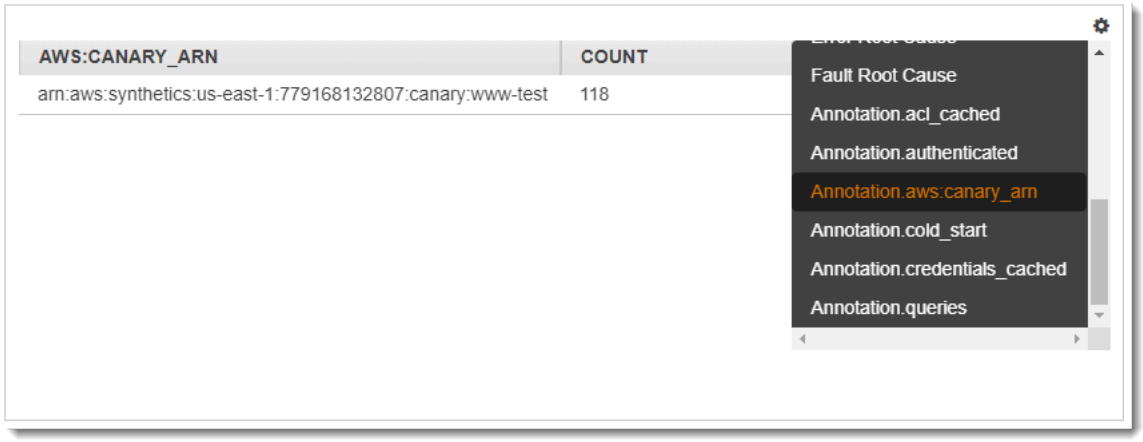
Die Ursache fortlaufender Fehler in Upstream- und Downstream-Services ermitteln

Sobald Sie einen CloudWatch Alarm für Fehler in einem Synthetics Canary erhalten, verwenden Sie die statistische Modellierung der Trace-Daten in X-Ray, um die wahrscheinliche Ursache des Problems in der X-Ray Analytics-Konsole zu ermitteln. In der Analytics-Konsole werden in der Tabelle mit den Ursachen der Antwortzeiten aufgezeichnete Entitätenpfade angezeigt. X-Ray bestimmt, welcher Pfad in Ihrer Spur die wahrscheinlichste Ursache für die Reaktionszeit ist. Das Format steht für eine Hierarchie von vorgefundenen Entitäten. Dies führt zu einer Reaktionszeit-Ursache.

Das folgende Beispiel zeigt, dass der Synthetics-Test für API „XXX“, der auf API Gateway ausgeführt wird, aufgrund einer Durchsatzkapazitätsausnahme aus der Amazon DynamoDB-Tabelle fehlschlägt.

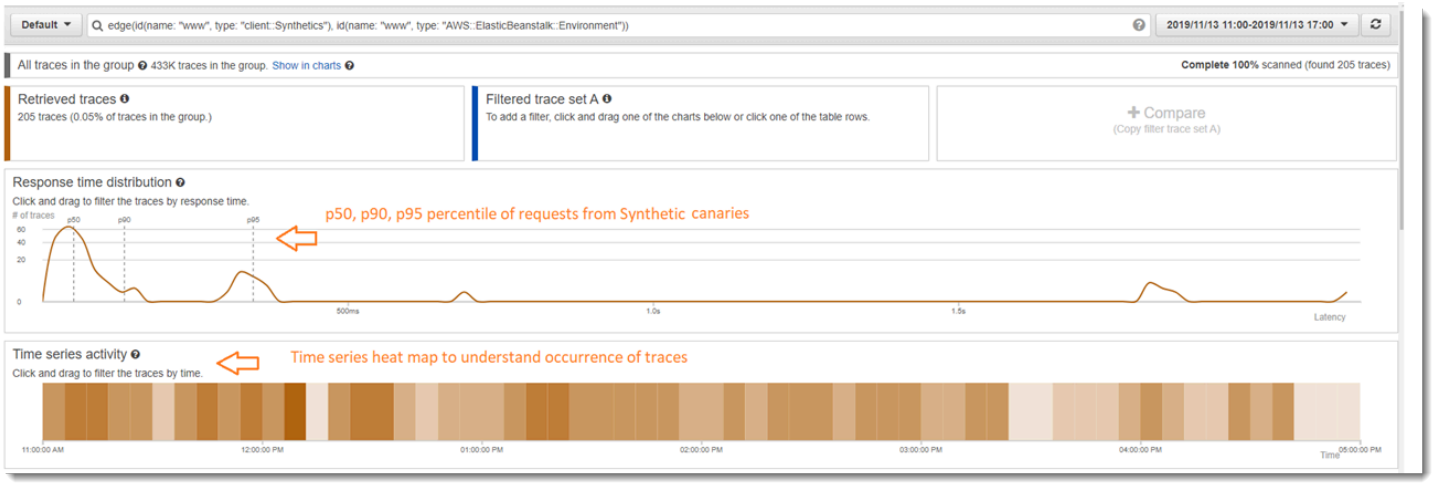

Canary
Map legend
Service details
Name: Canary
Type: client::Synthetics
View in Synthetics
Response distribution
Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.
100%
80%
60%
40%
20%
0%
0
200ms
400ms
600ms
800ms
1.0s
1.2s
Duration
Response status
Choose response 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



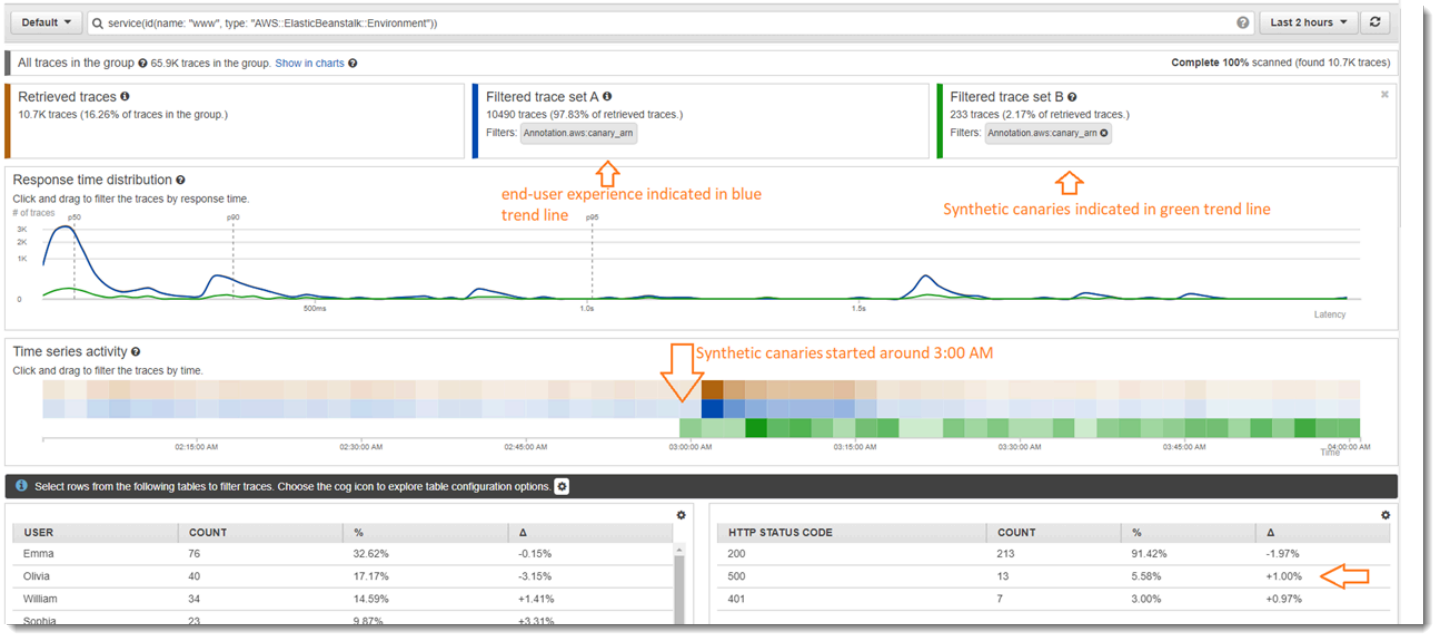
Performance-Engpässe und Trends identifizieren

Sie können Trends in der Leistung Ihres Endpunkts im Zeitverlauf anzeigen, indem Sie kontinuierlichen Datenverkehr von Ihren Synthetics-Kanaren verwenden, um eine Karte mit Trace-Details über einen bestimmten Zeitraum zu füllen.



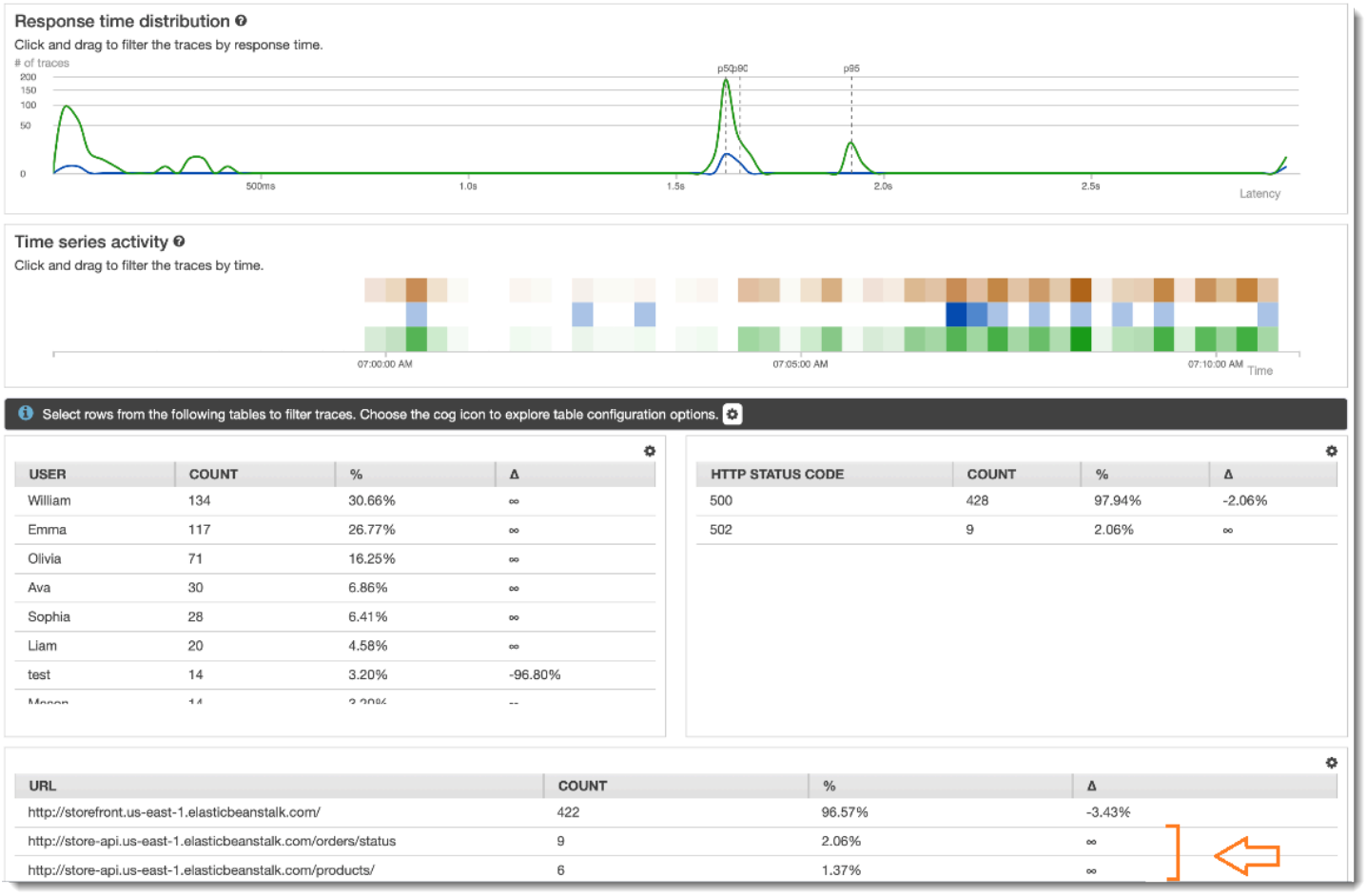
Latenz und Fehler- oder Ausfallraten vor und nach Änderungen vergleichen

Ermitteln Sie den Zeitpunkt, zu dem eine Änderung eingetreten ist, um diese Änderung mit einer Zunahme von Problemen zu korrelieren, die auf Ihren Kanaren aufgetreten sind. Verwenden Sie die X-Ray Analytics-Konsole, um die Zeitbereiche „Vorher“ und „Nachher“ als unterschiedliche Trace-Sets zu definieren und so eine visuelle Differenzierung in der Verteilung der Antwortzeiten zu erzielen.



Ermitteln Sie die erforderliche Kanarienabdeckung für alle APIs und URLs

Verwenden Sie X-Ray Analytics, um die Erfahrung von Canarys mit den Benutzern zu vergleichen. Die folgende Benutzeroberfläche zeigt eine blaue Trendlinie für Canarys und eine grüne Linie für Benutzer an. Sie können auch feststellen, dass bei zwei der drei URLs keine kanarischen Tests durchgeführt wurden.

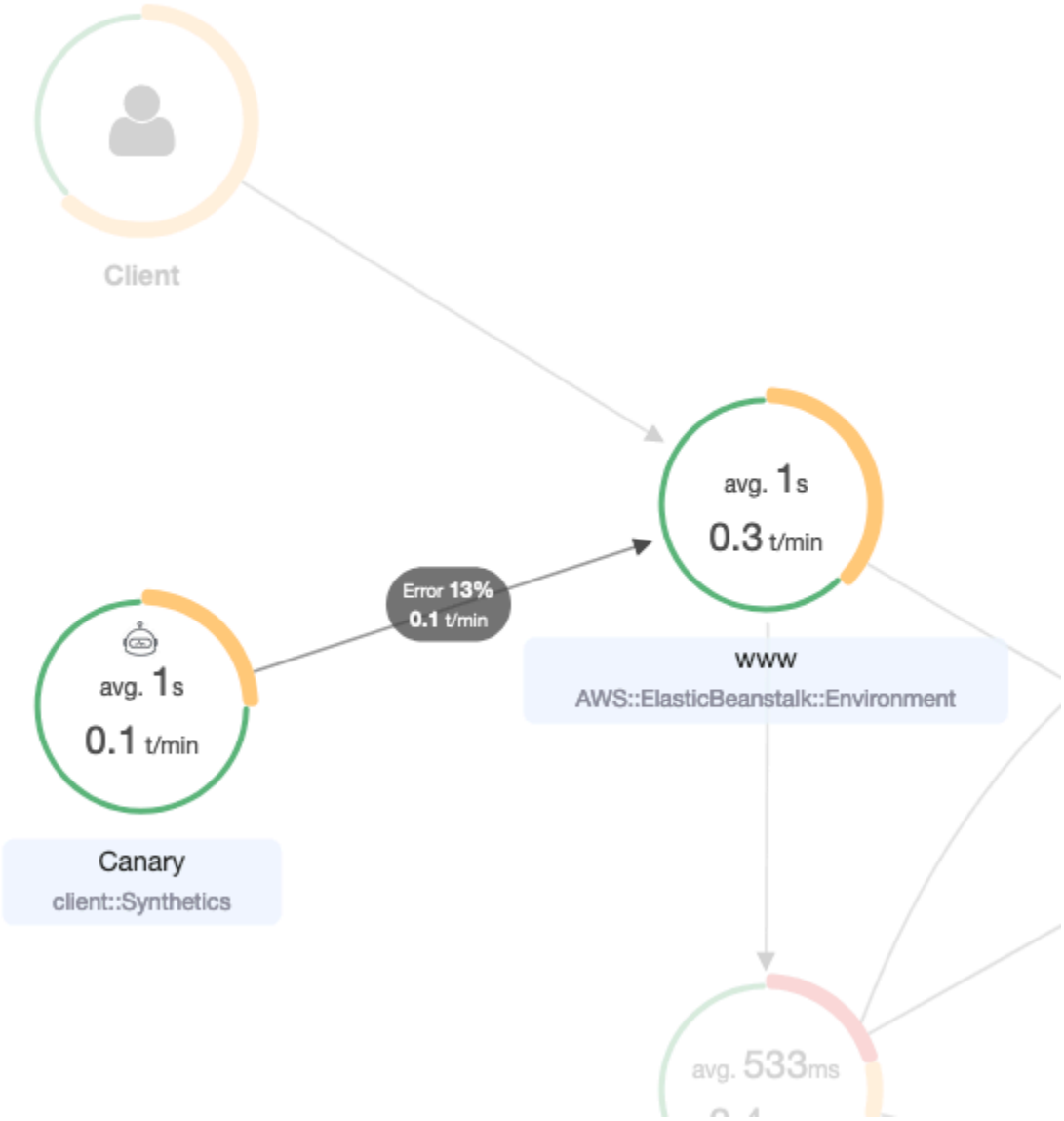


Gruppen für die Konzentration auf Synthetics-Tests verwenden

Sie können mithilfe eines Filterausdrucks eine X-Ray-Gruppe erstellen, um sich auf eine bestimmte Gruppe von Workflows zu konzentrieren, z. B. einen Synthetics für die Anwendung „www“, auf der AWS Elastic Beanstalk ausgeführt wird. Verwenden Sie die komplexen Schlüsselwörter `service()` und filtern `edge()` Sie nach Services und Edges.

Example Gruppenfilterausdruck

```
"edge(id(name: "www", type: "client::Synthetics"), id(name: "www", type:
 "AWS::ElasticBeanstalk::Environment"))"
```



AWS Elastic Beanstalk und AWS X-Ray

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon
> Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu

OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

AWS Elastic Beanstalk Zu den Plattformen gehört der X-Ray-Daemon. Sie können den Daemon ausführen, indem Sie eine Option in der Elastic Beanstalk Beanstalk-Konsole oder mit einer Konfigurationsdatei festlegen.

Auf der Java SE-Plattform können Sie eine Buildfile-Datei zur Erstellung Ihrer Anwendung mit einer Maven- oder Gradle-Instance verwenden. Das X-Ray-SDK SDK for Java ist bei Maven erhältlich, sodass Sie nur Ihren Anwendungscode bereitstellen und auf der Instanz erstellen können, um zu vermeiden, dass all Ihre Abhängigkeiten gebündelt und hochgeladen werden. AWS SDK für Java

Sie können die Elastic Beanstalk Beanstalk-Umgebungseigenschaften verwenden, um das X-Ray SDK zu konfigurieren. Die Methode, mit der Elastic Beanstalk Umgebungseigenschaften an Ihre Anwendung weitergibt, ist je nach Plattform unterschiedlich. Verwenden Sie je nach Plattform die Umgebungsvariablen oder Systemeigenschaften des X-Ray-SDK.

- Plattform Node.js — Verwenden Sie Umgebungsvariablen
- Java SE-Plattform — Verwenden Sie Umgebungsvariablen
- Tomcat-Plattform — Verwenden Sie Systemeigenschaften

Weitere Informationen finden Sie unter Configuring AWS X-Ray Debugging im AWS Elastic Beanstalk Developer Guide.

Elastic Load Balancing und AWS X-Ray

Elastic Load Balancing Application Load Balancer fügen eingehenden HTTP-Anfragen eine Trace-ID in einem Header mit dem Namen `X-Amzn-Trace-Id` hinzu.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Röntgen-Trace-ID-Format

Ein X-Ray `trace_id` besteht aus drei Zahlen, die durch Bindestriche getrennt sind. Beispiel, `1-58406520-a006649127e371903a2de979`. Dies umfasst:

- Die Versionsnummer, die ist. 1
- Die Uhrzeit der ursprünglichen Anfrage in Unix-Epochenzeit unter Verwendung von 8 Hexadezimalziffern.

  Zum Beispiel ist 10:00 Uhr am 1. Dezember 2016 PST in Epochenzeit `1480615200` Sekunden oder `58406520` in Hexadezimalziffern.
- Eine weltweit eindeutige 96-Bit-ID für den Trace mit 24 Hexadezimalziffern.

Load Balancer senden keine Daten an X-Ray und erscheinen nicht als Knoten auf Ihrer Service Map.

Weitere Informationen finden Sie unter Request Tracing for Your Application Load Balancer im Elastic Load Balancing Developer Guide.

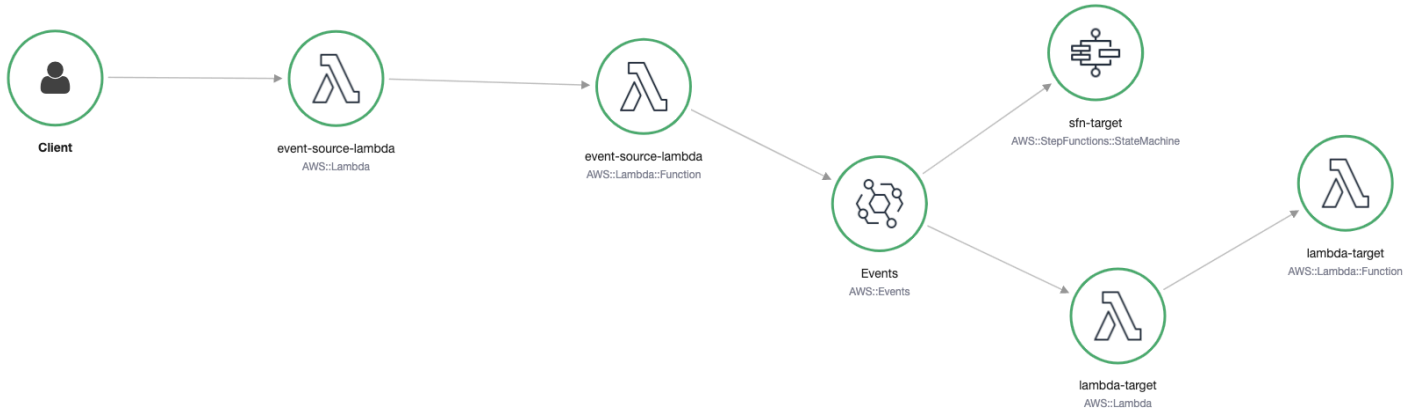
Amazon EventBridge und AWS X-Ray

AWS X-Ray lässt sich in Amazon integrieren EventBridge , um Ereignisse nachzuverfolgen, die durchlaufen wurden EventBridge. Wenn ein Dienst, der mit dem X-Ray SDK instrumentiert ist, Ereignisse sendet EventBridge, wird der Trace-Kontext an nachgeschaltete Ereignisziele innerhalb des Tracing-Headers weitergegeben. Das X-Ray SDK nimmt den Tracing-Header automatisch auf und wendet ihn auf alle nachfolgenden Instrumente an. Diese Kontinuität ermöglicht es Benutzern, alle nachgelagerten Dienste zu verfolgen, zu analysieren und zu debuggen, und bietet einen vollständigeren Überblick über ihr System.

Weitere Informationen finden Sie unter EventBridge X-Ray Integration im EventBridge Benutzerhandbuch.

Quelle und Ziele auf der X-Ray-Servicekarte anzeigen

Die X-Ray-Trace-Map zeigt einen EventBridge Ereignisknoten an, der Quell- und Zieldienste verbindet, wie im folgenden Beispiel:



Verbreiten Sie den Trace-Kontext an die Ereignisziele

Das X-Ray SDK ermöglicht es der EventBridge Ereignisquelle, den Trace-Kontext an nachgeschaltete Ereignisziele weiterzugeben. Die folgenden sprachspezifischen Beispiele demonstrieren den Aufruf EventBridge von einer Lambda-Funktion aus, für die die aktive Ablaufverfolgung aktiviert ist:

Java

Fügen Sie die erforderlichen Abhängigkeiten für X-Ray hinzu:

- AWS X-Ray SDK für Java
- AWS X-Ray Rekorder-SDK SDK for 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

Fügen Sie Ihrer Datei requirements.txt die folgende Abhängigkeit hinzu:

```
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#

Fügen Sie die folgenden X-Ray-Pakete zu Ihren C#-Abhängigkeiten hinzu:

```
<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 und AWS X-Ray

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können es verwenden AWS X-Ray , um Ihre AWS Lambda Funktionen zu verfolgen. Lambda führt den X-Ray-Daemon aus und zeichnet ein Segment mit Details zum Aufrufen und Ausführen der Funktion auf. Für weitere Instrumentierung können Sie das X-Ray SDK mit Ihrer Funktion bündeln, um ausgehende Anrufe aufzuzeichnen und Anmerkungen und Metadaten hinzuzufügen.

Wenn Ihre Lambda-Funktion von einem anderen instrumentierten Dienst aufgerufen wird, verfolgt Lambda Anfragen, die bereits ohne zusätzliche Konfiguration abgetastet wurden. Der Upstream-Service kann eine instrumentierte Webanwendung oder eine andere Lambda-Funktion sein. Ihr Service kann die Funktion direkt mit einem instrumentierten AWS SDK-Client oder durch Aufrufen einer API Gateway mit einem instrumentierten HTTP-Client aufrufen.

AWS X-Ray unterstützt die Ablaufverfolgung ereignisgesteuerter Anwendungen mithilfe von Amazon AWS Lambda SQS. Verwenden Sie die CloudWatch Konsole, um eine verbundene Ansicht jeder Anfrage zu sehen, während sie bei Amazon SQS in die Warteschlange gestellt und von einer nachgeschalteten Lambda-Funktion verarbeitet wird. Traces von Upstream-Nachrichtenproduzenten werden automatisch mit Traces von nachgeschalteten Lambda-Consumer-Knoten verknüpft, wodurch eine end-to-end Ansicht der Anwendung erstellt wird. Weitere Informationen finden Sie unter Ablaufverfolgung ereignisgesteuerter Anwendungen.

> **Note**
>
> Wenn Sie Traces für eine Downstream-Lambda-Funktion aktiviert haben, müssen Sie auch Traces für die Root-Lambda-Funktion aktiviert haben, die die Downstream-Funktion aufruft, damit die Downstream-Funktion Traces generiert.

Wenn Ihre Lambda-Funktion nach einem Zeitplan ausgeführt wird oder von einem Dienst aufgerufen wird, der nicht instrumentiert ist, können Sie Lambda so konfigurieren, dass es Aufrufe mit aktiver Ablaufverfolgung abtastet und aufzeichnet.

So konfigurieren Sie die X-Ray-Integration für eine AWS Lambda Funktion

1. Öffnen Sie die AWS Lambda -Konsole.
2. Wählen Sie in der linken Navigationsleiste Funktionen aus.
3. Wählen Sie Ihre Funktion.
4. Scrollen Sie auf der Registerkarte Konfiguration nach unten zur Karte Zusätzliche Überwachungstools. Sie finden diese Karte auch, indem Sie im linken Navigationsbereich die Option Überwachungs- und Betriebstools auswählen.
5. Wählen Sie Bearbeiten aus.
6. Aktivieren Sie unter AWS X-Ray die Option Aktive Ablaufverfolgung.

Bei Laufzeiten mit einem entsprechenden X-Ray-SDK führt Lambda auch den X-Ray-Daemon aus.

X-Ray SDKs auf Lambda

- X-Ray SDK for Go — Go 1.7 und neuere Laufzeiten
- X-Ray SDK for Java — Java 8-Laufzeit
- X-Ray SDK für Node.js — Node.js 4.3 und neuere Laufzeiten
- X-Ray SDK für Python — Python 2.7, Python 3.6 und neuere Laufzeiten
- X-Ray SDK for .NET — .NET Core 2.0 und neuere Laufzeiten

Um das X-Ray SDK auf Lambda zu verwenden, bündeln Sie es jedes Mal mit Ihrem Funktionscode, wenn Sie eine neue Version erstellen. Sie können Ihre Lambda-Funktionen mit denselben Methoden instrumentieren, die Sie zur Instrumentierung von Anwendungen verwenden, die auf anderen Diensten ausgeführt werden. Der Hauptunterschied besteht darin, dass Sie nicht das SDK dazu verwenden, eingehende Anforderungen zu instrumentieren, Samplingentscheidungen zu treffen und Segmente zu erstellen.

Der andere Unterschied zwischen der Instrumentierung von Lambda-Funktionen und Webanwendungen besteht darin, dass das Segment, das Lambda erstellt und an X-Ray sendet, nicht durch Ihren Funktionscode geändert werden kann. Sie können Untersegmente erstellen und Anmerkungen und Metadaten aufzeichnen. Sie können dem übergeordneten Segment jedoch keine Anmerkungen und Metadaten hinzufügen.

Weitere Informationen finden Sie unter Verwenden von AWS X-Ray im AWS Lambda Entwicklerhandbuch.

AWS Step Functions und AWS X-Ray

AWS X-Ray lässt sich integrieren AWS Step Functions , um Anfragen für Step Functions zu verfolgen und zu analysieren. Sie können die Komponenten Ihrer Zustandsmaschine visualisieren, Leistungsengpässe identifizieren und Anfragen, die zu einem Fehler geführt haben, beheben. Weitere Informationen finden Sie unter AWS X-Ray und Step Functions im AWS Step Functions Entwicklerhandbuch.

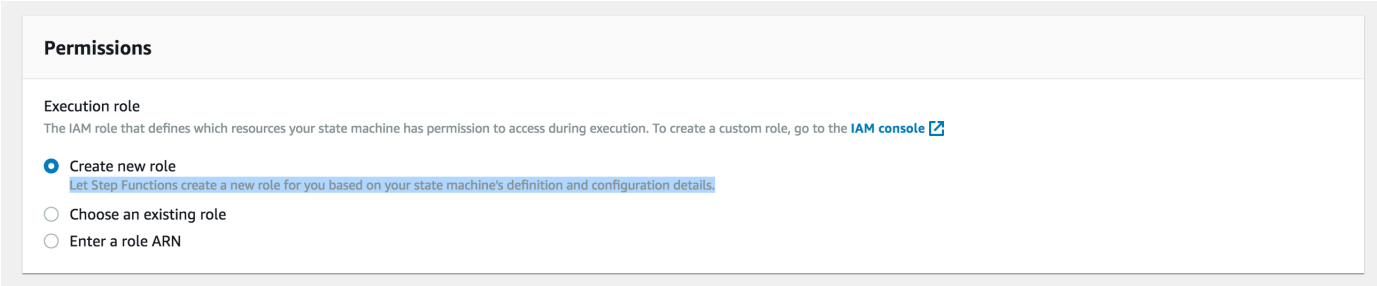
Um X-Ray Tracing beim Erstellen einer neuen State Machine zu aktivieren

1. Öffnen Sie die Step Functions Functions-Konsole unter https://console.aws.amazon.com/states/.
2. Wählen Sie Create a State Machine aus.

3. Wählen Sie auf der Seite „Zustandsmaschine definieren“ entweder Mit Codefragmenten verfassen oder Mit einer Vorlage beginnen aus. Wenn Sie ein Beispielprojekt ausführen möchten, können Sie die Röntgenverfolgung während der Erstellung nicht aktivieren. Aktivieren Sie stattdessen die Röntgenverfolgung, nachdem Sie Ihre Zustandsmaschine erstellt haben.
4. Wählen Sie Weiter.
5. Konfigurieren Sie auf der Seite „Details angeben“ Ihren Zustandsmaschine.
6. Wählen Sie X-Ray Tracing aktivieren.

Um X-Ray Tracing in einer vorhandenen Zustandsmaschine zu aktivieren

1. Wählen Sie in der Step Functions Functions-Konsole den Zustandsmaschine aus, für den Sie die Ablaufverfolgung aktivieren möchten.
2. Wählen Sie Edit (Bearbeiten) aus.
3. Wählen Sie X-Ray Tracing aktivieren.
4. (Optional) Generieren Sie automatisch eine neue Rolle für Ihren Zustandsmaschine mit X-Ray-Berechtigungen, indem Sie im Fenster „Berechtigungen“ die Option Neue Rolle erstellen auswählen.



5. Wählen Sie Save (Speichern) aus.

> **Note**
>
> Wenn Sie eine neue Zustandsmaschine erstellen, wird sie automatisch verfolgt, wenn die Anfrage gesampelt wird und die Ablaufverfolgung in einem Upstream-Service wie Amazon API Gateway oder aktiviert ist. AWS LambdaÜberprüfen Sie für alle vorhandenen Zustandsmaschinen, die nicht über die Konsole konfiguriert wurden, z. B. über eine CloudFormation Vorlage, ob Sie über eine IAM-Richtlinie verfügen, die ausreichende Berechtigungen zum Aktivieren von X-Ray-Traces gewährt.

Instrumentierung Ihrer Anwendung für AWS X-Ray

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Die Instrumentierung Ihrer Anwendung umfasst das Senden von Trace-Daten für eingehende und ausgehende Anfragen und andere Ereignisse innerhalb Ihrer Anwendung zusammen mit Metadaten zu jeder Anfrage. Es gibt verschiedene Instrumentierungsoptionen, aus denen Sie je nach Ihren speziellen Anforderungen wählen oder diese kombinieren können:

- auto Instrumentierung — Instrumentierung Ihrer Anwendung ohne Codeänderungen, typischerweise durch Konfigurationsänderungen, Hinzufügen eines Agenten für automatische Instrumentierung oder andere Mechanismen.
- Bibliotheksinstrumentierung — Nehmen Sie minimale Änderungen am Anwendungscode vor, um vorgefertigte Instrumentierung hinzuzufügen, die auf bestimmte Bibliotheken oder Frameworks wie das AWS SDK, Apache HTTP-Clients oder SQL-Clients abzielt.
- Manuelle Instrumentierung — fügen Sie Ihrer Anwendung an jeder Stelle, an die Sie Trace-Informationen senden möchten, Instrumentierungscode hinzu.

Es gibt mehrere SDKs Agenten und Tools, die verwendet werden können, um Ihre Anwendung für die Röntgenverfolgung zu instrumentieren.

Topics

Instrumentierung Ihrer Anwendung mit der Distro für AWS OpenTelemetry

The AWS Distro for OpenTelemetry (ADOT) ist eine AWS Distribution, die auf dem Projekt Cloud Native Computing Foundation (CNCF) basiert. OpenTelemetry OpenTelemetry bietet einen einzigen Satz von Open-Source-Dateien APIs, Bibliotheken und Agenten zur Erfassung verteilter Traces und Metriken. Dieses Toolkit ist eine Distribution von OpenTelemetry Upstream-Komponenten SDKs, einschließlich Agenten für automatische Instrumentierung und Kollektoren, die von getestet, optimiert, gesichert und unterstützt werden. AWS

Mit ADOT können Techniker ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Überwachungslösungen wie Amazon CloudWatch und Amazon AWS X-Ray OpenSearch Service senden.

Die Verwendung von X-Ray mit ADOT erfordert zwei Komponenten: ein OpenTelemetry SDK, das für die Verwendung mit X-Ray aktiviert ist, und das AWS Distro for OpenTelemetry Collector, das für die Verwendung mit X-Ray aktiviert ist. Weitere Informationen zur Verwendung der AWS Distribution für OpenTelemetry AWS X-Ray und andere finden Sie in der AWS-Services Dokumentation zur AWS Distribution. OpenTelemetry

Weitere Informationen zur Sprachunterstützung und -verwendung finden Sie unter AWS Observability on. GitHub

> **Note**
>
> Sie können den CloudWatch Agenten jetzt verwenden, um Metriken, Protokolle und Traces von EC2 Amazon-Instances und lokalen Servern zu sammeln. CloudWatch Agentenversion 1.300025.0 und höher können Traces von OpenTelemetryunserem X-Ray-Client SDKs sammeln und an X-Ray senden. Wenn Sie den CloudWatch Agenten anstelle des AWS Distro for OpenTelemetry (ADOT) Collector oder des X-Ray-Daemons zum Sammeln von Traces verwenden, können Sie die Anzahl der verwalteten Agenten reduzieren. Weitere Informationen finden Sie unter dem Thema CloudWatch Agent im CloudWatch Benutzerhandbuch.

ADOT umfasst Folgendes:

- AWS Distro für Go OpenTelemetry

- AWS Distro für Java OpenTelemetry
- AWS Distro für OpenTelemetry JavaScript
- AWS Distro für Python OpenTelemetry
- AWS Distro für .NET OpenTelemetry

ADOT bietet derzeit Unterstützung für automatische Instrumentierung für Java und Python. Darüber hinaus ermöglicht ADOT die automatische Instrumentierung von AWS Lambda-Funktionen und ihren Downstream-Anfragen mithilfe von Java-, Node.js- und Python-Laufzeiten über ADOT Managed Lambda Layers.

ADOT SDKs für Java und Go unterstützt zentralisierte X-Ray-Probenahmeregeln. Wenn Sie Unterstützung für X-Ray-Sampling-Regeln in anderen Sprachen benötigen, sollten Sie die Verwendung eines AWS X-Ray SDK in Betracht ziehen.

> **Note**
>
> Sie können jetzt W3C-Trace an X-Ray IDs senden. Standardmäßig OpenTelemetry haben Traces, die mit erstellt wurden, ein Trace-ID-Format, das auf der W3C Trace Context-Spezifikation basiert. Dies unterscheidet sich von dem Format für Traces IDs , die mit einem X-Ray-SDK oder durch AWS Dienste, die in X-Ray integriert sind, erstellt werden. Um sicherzustellen, dass Trace IDs im W3C-Format von X-Ray akzeptiert wird, müssen Sie AWS X-Ray Exporter Version 0.86.0 oder höher verwenden, die in ADOT Collector Version 0.34.0 und höher enthalten ist. Frühere Versionen des Exporters validieren Trace-ID-Zeitstempel, was dazu führen kann, dass W3C-Trace abgelehnt wird. IDs

Instrumentieren Sie Ihre Anwendung mit AWS X-Ray SDKs

AWS X-Ray enthält eine Reihe von sprachspezifischen SDKs Funktionen, mit denen Sie Ihre Anwendung so einrichten können, dass sie Traces an X-Ray sendet. Jedes X-Ray-SDK bietet Folgendes:

- Interceptors, die Sie Ihrem Code hinzufügen können, um eingehende HTTP-Anforderungen nachzuverfolgen.
- Client-Handler zur Instrumentierung von AWS SDK-Clients, die Ihre Anwendung verwendet, um andere aufzurufen AWS-Services

- Ein HTTP-Client zur Instrumentierung von Aufrufen an andere interne und externe HTTP-Webdienste

X-Ray unterstützt SDKs auch Instrumentierung von Aufrufen von SQL-Datenbanken, automatische AWS SDK-Client-Instrumentierung und andere Funktionen. Anstatt Trace-Daten direkt an X-Ray zu senden, sendet das SDK JSON-Segmentdokumente an einen Daemon-Prozess, der auf UDP-Verkehr wartet. Der X-Ray-Daemon puffert Segmente in einer Warteschlange und lädt sie stapelweise auf X-Ray hoch.

Die folgenden sprachspezifischen Optionen sind verfügbar: SDKs

- AWS X-Ray SDK for Go
- AWS X-Ray SDK für Java
- AWS X-Ray SDK für Node.js
- AWS X-Ray SDK für Python
- AWS X-Ray SDK for .NET
- AWS X-Ray SDK for Ruby

X-Ray bietet derzeit Unterstützung für automatische Instrumentierung für Java.

Wahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs

Die SDKs im Lieferumfang von X-Ray enthaltenen Geräte sind Teil einer eng integrierten Instrumentierungslösung, die von angeboten wird AWS. Das AWS Distro for OpenTelemetry ist Teil einer umfassenderen Branchenlösung, bei der X-Ray nur eine von vielen Tracing-Lösungen ist. Sie können die end-to-end Ablaufverfolgung in X-Ray mit beiden Methoden implementieren, aber es ist wichtig, die Unterschiede zu verstehen, um den für Sie nützlichsten Ansatz zu finden.

Wir empfehlen, Ihre Anwendung mit der AWS Distribution zu instrumentieren, OpenTelemetry wenn Sie Folgendes benötigen:

- Die Möglichkeit, Traces an mehrere verschiedene Tracing-Backends zu senden, ohne Ihren Code erneut instrumentieren zu müssen
- Support für eine große Anzahl von Bibliotheksinstrumenten für jede Sprache, verwaltet von der Community OpenTelemetry

- Vollständig verwaltete Lambda-Schichten, die alles zusammenfassen, was Sie für die Erfassung von Telemetriedaten benötigen, ohne dass Codeänderungen erforderlich sind, wenn Sie Java, Python oder Node.js verwenden

> **Note**
>
> AWS Distro for OpenTelemetry bietet einen einfacheren Einstieg in die Instrumentierung Ihrer Lambda-Funktionen. Aufgrund der gebotenen Flexibilität OpenTelemetry benötigt Ihre Lambda-Funktion jedoch zusätzlichen Speicher, und bei Aufrufen kann es zu einer Erhöhung der Kaltstart-Latenz kommen, was zu zusätzlichen Kosten führen kann. Wenn Sie für niedrige Latenzzeiten optimieren und keine erweiterten Funktionen wie dynamisch konfigurierbare Back-End-Ziele benötigen OpenTelemetry, sollten Sie das AWS X-Ray-SDK zur Instrumentierung Ihrer Anwendung verwenden.

Wir empfehlen, ein X-Ray-SDK für die Instrumentierung Ihrer Anwendung zu wählen, wenn Sie Folgendes benötigen:

- Eine eng integrierte Lösung aus einer Hand
- Integration mit zentralisierten X-Ray-Probenregeln, einschließlich der Möglichkeit, Sampling-Regeln von der X-Ray-Konsole aus zu konfigurieren und sie automatisch auf mehreren Hosts zu verwenden, wenn Node.js, Python, Ruby oder .NET verwendet werden

Transaktionssuche

Transaktionssuche ist ein interaktives Analyseerlebnis, mit dem Sie sich einen vollständigen Überblick über die Transaktionsspannen Ihrer Anwendung verschaffen können. Spans sind die grundlegenden Vorgangseinheiten in einer verteilten Ablaufverfolgung und stellen bestimmte Aktionen oder Aufgaben in einer Anwendung oder einem System dar. Jeder Span zeichnet Details zu einem bestimmten Segment der Transaktion auf. Zu diesen Details gehören Start- und Endzeiten, Dauer und zugehörige Metadaten, zu denen auch Geschäftsattribute wie Kunde IDs und Bestellung gehören können IDs. Die Spans sind in einer Hierarchie mit über- und untergeordneten Elementen angeordnet. Diese Hierarchie bildet eine vollständige Ablaufverfolgung, die den Ablauf einer Transaktion über verschiedene Komponenten oder Dienste hinweg abbildet.

Weitere Informationen finden Sie unter Transaktionssuche.

OpenTelemetry Protokollendpunkt (OTLP)

OpenTelemetry ist ein Open-Source-Observability-Framework, das IT-Teams standardisierte Protokolle und Tools für die Erfassung und Weiterleitung von Telemetriedaten zur Verfügung stellt. Es bietet ein einheitliches Format für die Instrumentierung, die Generierung, die Erfassung und den Export von Anwendungstelemetriedaten wie Metriken, Protokollen und Traces in Überwachungsplattformen, um Analysen zu erstellen und Erkenntnisse zu gewinnen. Durch die Verwendung OpenTelemetry können Teams eine Abhängigkeit von einem Anbieter vermeiden und so die Flexibilität ihrer Observability-Lösungen gewährleisten.

Sie können OpenTelemetry damit Traces direkt an einen OpenTelemetry Protokoll-Endpunkt (OTLP) senden und in CloudWatch Application Signals umfassende Erfahrungen out-of-the mit der Überwachung der Anwendungsleistung abrufen.

Weitere Informationen finden Sie unter OpenTelemetry.

Verwenden von Go

Es gibt zwei Möglichkeiten, Ihre Go-Anwendung so zu instrumentieren, dass sie Spuren an X-Ray sendet:

- AWS Distro for OpenTelemetry Go — Eine AWS Distribution, die eine Reihe von Open-Source-Bibliotheken zum Senden korrelierter Metriken und Traces über AWS Distro for Collector an mehrere AWS Überwachungslösungen wie Amazon CloudWatch und Amazon OpenSearch Service bereitstellt. AWS X-Ray OpenTelemetry
- AWS X-Ray SDK for Go — Eine Reihe von Bibliotheken zum Generieren und Senden von Traces an X-Ray über den X-Ray-Daemon.

Weitere Informationen finden Sie unter Wahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs.

AWS Distro für Go OpenTelemetry

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Mit AWS Distro for OpenTelemetry Go können Sie Ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Überwachungslösungen wie Amazon CloudWatch AWS X-Ray, und Amazon OpenSearch Service senden. Die Verwendung von X-Ray mit AWS Distro for OpenTelemetry erfordert zwei Komponenten: ein OpenTelemetry SDK, das für die Verwendung mit X-Ray aktiviert ist, und das AWS Distro for OpenTelemetry Collector, das für die Verwendung mit X-Ray aktiviert ist.

Informationen zu den ersten Schritten finden Sie in der Dokumentation zu AWS Distro for OpenTelemetry Go.

Weitere Informationen zur Verwendung von AWS Distro for OpenTelemetry with AWS X-Ray und anderen AWS-Services finden Sie unter AWS Distro for OpenTelemetry oder Distro for Documentation AWS . OpenTelemetry

Weitere Informationen zur Sprachunterstützung und -verwendung finden Sie unter AWS Observability on. GitHub

AWS X-Ray SDK for Go

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK SDK for Go besteht aus einer Reihe von Bibliotheken für Go-Anwendungen, die Klassen und Methoden zum Generieren und Senden von Trace-Daten an den X-Ray-Daemon bereitstellen. Zu den Trace-Daten gehören Informationen über eingehende HTTP-Anfragen, die von der Anwendung bedient werden, sowie über Aufrufe, die die Anwendung mithilfe des AWS SDK, HTTP-Clients oder eines SQL-Datenbank-Connectors an Downstream-Dienste sendet. Sie können Segmente auch manuell erstellen und Debug-Informationen Anmerkungen und Metadaten hinzufügen.

Laden Sie das SDK aus seinem GitHubRepository herunter mit`go get`:

```
$ go get -u github.com/aws/aws-xray-sdk-go/...
```

Für Webanwendungen beginnen Sie damit, die Funktion `xray.Handler` zu verwenden, um eingehende Anforderungen nachzuverfolgen. Der Message Handler erstellt für jede rückverfolgte

Anforderung ein Segment und vervollständigt das Segment, nachdem die Antwort gesendet wurde. Während das Segment geöffnet ist, können Sie die SDK-Client-Methoden nutzen, um dem Segment Informationen hinzuzufügen, Untersegmente zu erstellen und nachgelagerte Aufrufe rückzuverfolgen. Das SDK erfasst auch automatisch Ausnahmen, die Ihre Anwendung ausgibt, während das Segment geöffnet ist.

Bei Lambda-Funktionen, die von einer instrumentierten Anwendung oder einem Dienst aufgerufen werden, liest Lambda den Tracing-Header und verfolgt automatisch Sampling-Anfragen. Für andere Funktionen können Sie Lambda so konfigurieren, dass eingehende Anfragen abgefragt und verfolgt werden. In beiden Fällen erstellt Lambda das Segment und stellt es dem X-Ray SDK zur Verfügung.

> **Note**
>
> Auf Lambda ist das X-Ray SDK optional. Wenn Sie es nicht in Ihrer Funktion verwenden, enthält Ihre Service-Map immer noch einen Knoten für den Lambda-Service und einen für jede Lambda-Funktion. Durch Hinzufügen des SDK können Sie Ihren Funktionscode instrumentieren, um Untersegmente zu dem von Lambda aufgezeichneten Funktionssegment hinzuzufügen. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Als Nächstes umschließen Sie den Client mit einem Aufruf der `AWS`-Funktion. Dieser Schritt stellt sicher, dass X-Ray Aufrufe jeder Client-Methode instrumentiert. Sie können auch Aufrufe von SQL-Datenbanken instrumentieren.

Nachdem Sie das SDK verwendet haben, passen Sie sein Verhalten an, indem Sie den Rekorder und die Middleware konfigurieren. Sie können Plugins zum Festhalten von Daten über die Datenverarbeitungsressourcen, auf denen Ihre Anwendung ausgeführt wird, hinzufügen, das Samplingverhalten durch Samplingregeln anpassen und Protokollebenen einrichten, um mehr oder weniger Informationen von dem SDK in Ihren Anwendungsprotokollen zu sehen.

Zeichnen Sie zusätzliche Informationen zu Anforderungen und den Aufgaben, die Ihre Anwendung ausführt, in Anmerkungen und Metadaten auf. Anmerkungen sind einfache Schlüsselwertpaare, die für die Verwendung mit Filterausdrücken indiziert werden, damit Sie nach Ablaufverfolgen mit bestimmten Daten suchen können. Metadateneinträge sind weniger einschränkend und können ganze Objekte und Arrays aufzeichnen – alle Daten, die in eine JSON zusammengefasst werden können.

> **Anmerkungen und Metadaten**
>
> Anmerkungen und Metadaten sind beliebiger Text, den Sie Segmenten mit dem X-Ray SDK hinzufügen. Anmerkungen werden für die Verwendung mit Filterausdrücken indexiert. Metadaten werden nicht indexiert, können aber im Rohsegment mit der X-Ray-Konsole oder API angezeigt werden. Jeder, dem Sie Lesezugriff auf X-Ray gewähren, kann diese Daten einsehen.

Wenn Sie viele instrumentierten Clients in Ihrem Code haben, kann ein einzelnes Anforderungssegmente viele Untersegmente enthalten, eines für jeden Aufruf mit einem instrumentierten Client. Sie können Untersegmente organisieren und gruppieren, indem Sie Client-Aufrufe in benutzerdefinierten Untersegmenten zusammenfassen. Sie können ein benutzerdefiniertes Untersegment für eine ganze Funktion oder eine Code-Abschnitt erstellen und Metadaten und Anmerkungen im Untersegment festhalten, anstatt alles im übergeordneten Segment aufzuzeichnen.

Voraussetzungen

Das X-Ray SDK for Go benötigt Go 1.9 oder höher.

Das SDK hängt von den folgenden Bibliotheken zur Kompilierungs- und Laufzeit ab:

- AWS SDK for Go Version 1.10.0 oder neuer

Diese Abhängigkeiten werden in der `README.md`-Datei des SDK angegeben.

Referenzdokumentation

Nachdem Sie das SDK heruntergeladen haben, erstellen und hosten Sie die Dokumentation lokal, um sie in einem Webbrowser anzuzeigen.

So zeigen Sie die Referenzdokumentation an

1. Navigieren Sie zum (Linux- oder Mac-)Verzeichnis `$GOPATH/src/github.com/aws/aws-xray-sdk-go` oder zum (Windows-)Ordner `%GOPATH%\src\github.com\aws\aws-xray-sdk-go`
2. Führen Sie den Befehl `godoc` aus.

```
$ godoc -http=:6060
```

3. Öffnen Sie einen Browser unter `http://localhost:6060/pkg/github.com/aws/aws-xray-sdk-go/`.

Konfigurieren des X-Ray-SDK für Go

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können die Konfiguration für das X-Ray SDK for Go über Umgebungsvariablen angeben, indem Sie sie `Configure` mit einem `Config` Objekt aufrufen oder Standardwerte annehmen. Umgebungsvariablen haben Vorrang vor `Config`-Werten, die wiederum Vorrang vor Standardwerten haben.

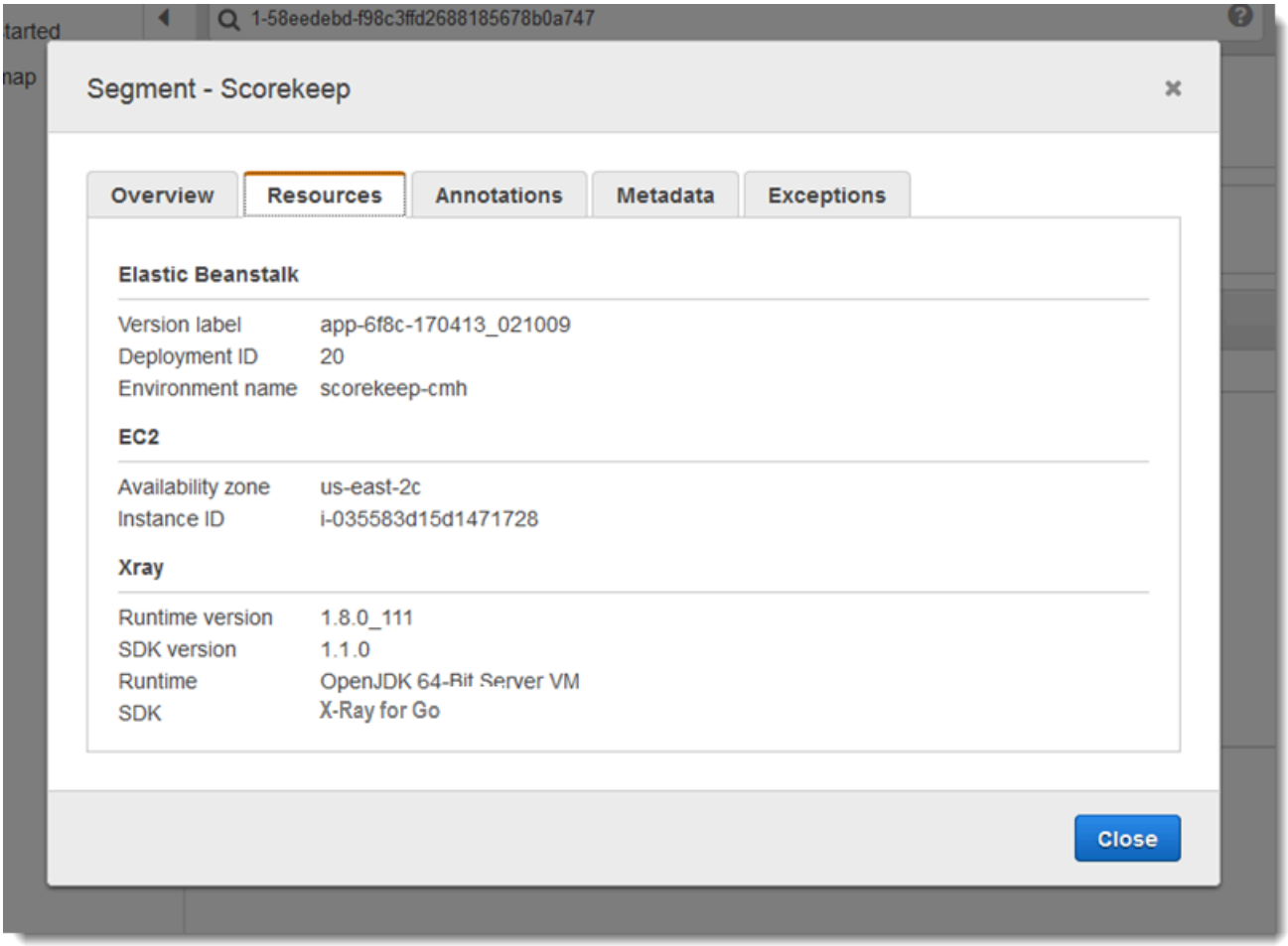
Sections

Service-Plugins

Wird verwendet`plugins`, um Informationen über den Dienst aufzuzeichnen, der Ihre Anwendung hostet.

Plugins

- Amazon EC2 — `EC2Plugin` fügt die Instance-ID, die Availability Zone und die CloudWatch Logs-Gruppe hinzu.
- Elastic Beanstalk — `ElasticBeanstalkPlugin` fügt den Umgebungsnamen, die Versionsbezeichnung und die Bereitstellungs-ID hinzu.
- Amazon ECS — `ECSPlugin` fügt die Container-ID hinzu.



Um ein Plugin zu verwenden, importieren Sie eines der folgenden Pakete.

```
"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"
```

Jedes Plugin hat einen expliziten `Init()`-Funktionsaufruf, der das Plugin lädt.

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",
  })
}
```

Das SDK verwendet auch Plugin-Einstellungen, um das `origin` Feld für das Segment festzulegen. Dies gibt den AWS Ressourcentyp an, auf dem Ihre Anwendung ausgeführt wird. Wenn Sie mehrere Plugins verwenden, verwendet das SDK die folgende Auflösungsreihenfolge, um den Ursprung zu bestimmen: ElasticBeanstalk > EKS > ECS > EC2.

Regeln für die Probenahme

Das SDK verwendet die Sampling-Regeln, die Sie in der X-Ray-Konsole definieren, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen. Die Standardregel verfolgt die erste Anfrage jede Sekunde und fünf Prozent aller weiteren Anfragen aller Dienste, die Traces an X-Ray senden. Erstellen Sie zusätzliche Regeln in der X-Ray-Konsole, um die Menge der aufgezeichneten Daten für jede Ihrer Anwendungen anzupassen.

Das SDK wendet benutzerdefinierte Regeln in der Reihenfolge an, in der sie definiert sind. Wenn eine Anfrage mehreren benutzerdefinierten Regeln entspricht, wendet das SDK nur die erste Regel an.

> **Note**
>
> Wenn das SDK X-Ray nicht erreichen kann, um Sampling-Regeln abzurufen, kehrt es zu einer lokalen Standardregel zurück, die die erste Anfrage pro Sekunde und fünf Prozent aller zusätzlichen Anfragen pro Host vorsieht. Dies kann passieren, wenn der Host nicht berechtigt

ist APIs, Sampling aufzurufen, oder wenn er keine Verbindung zum X-Ray-Daemon herstellen kann, der als TCP-Proxy für API-Aufrufe durch das SDK fungiert.

Sie können das SDK auch so konfigurieren, dass Sampling-Regeln aus einem JSON-Dokument geladen werden. Das SDK kann lokale Regeln als Backup für Fälle verwenden, in denen X-Ray Sampling nicht verfügbar ist, oder ausschließlich lokale Regeln verwenden.

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
  }
}
```

In diesem Beispiel werden eine benutzerdefinierte Regel und eine Standardregel definiert. Die benutzerdefinierte Regel wendet eine Stichprobenrate von fünf Prozent an, ohne dass eine Mindestanzahl von Anfragen für Pfade verfolgt werden muss. `/api/move/` Die Standardregel verfolgt die erste Anfrage jede Sekunde und 10 Prozent der weiteren Anfragen.

Der Nachteil der lokalen Definition von Regeln besteht darin, dass das feste Ziel von jeder Instanz des Rekorders unabhängig angewendet wird, anstatt vom X-Ray-Dienst verwaltet zu werden. Wenn Sie mehr Hosts bereitstellen, wird die feste Rate vervielfacht, wodurch es schwieriger wird, die Menge der aufgezeichneten Daten zu kontrollieren.

Wenn aktiviert AWS Lambda, können Sie die Samplerate nicht ändern. Wenn Ihre Funktion von einem instrumentierten Dienst aufgerufen wird, werden Aufrufe, die Anfragen generierten, die von

diesem Dienst abgetastet wurden, von Lambda aufgezeichnet. Wenn aktives Tracing aktiviert ist und kein Tracing-Header vorhanden ist, trifft Lambda die Stichprobenentscheidung.

Um Sicherungsregeln bereitzustellen, verweisen Sie mit `NewCentralizedStrategyWithFilePath` auf die lokale Sampling-JSON-Datei.

Example main.go — Lokale Stichprobenregel

```
s, _ := sampling.NewCentralizedStrategyWithFilePath("sampling.json") // path to local
 sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Um nur lokale Regeln zu verwenden, verweisen Sie mit `NewLocalizedStrategyFromFilePath` auf die lokale Sampling-JSON-Datei.

Example main.go — Sampling deaktivieren

```
s, _ := sampling.NewLocalizedStrategyFromFilePath("sampling.json") // path to local
 sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Protokollierung

> **Note**
>
> Die `xray.Config{}`-Felder `LogLevel` und `LogFormat` sind ab Version 1.0.0-rc.10 veraltet.

X-Ray verwendet die folgende Schnittstelle für die Protokollierung. Der Standardlogger schreibt in `stdout` bei `LogLevelInfo` und höher.

```
type Logger interface {
 Log(level LogLevel, msg fmt.Stringer)
}

const (
 LogLevelDebug LogLevel = iota + 1
 LogLevelInfo
 LogLevelWarn
 LogLevelError
```

```
)
```

Example Schreibvorgänge in **`io.Writer`**

```
xray.SetLogger(xraylog.NewDefaultLogger(os.Stderr, xraylog.LogLevelError))
```

Umgebungsvariablen

Sie können Umgebungsvariablen verwenden, um das X-Ray SDK for Go zu konfigurieren. Das SDK unterstützt die folgenden Variablen.

- `AWS_XRAY_CONTEXT_MISSING`— Stellen Sie diese Option ein`RUNTIME_ERROR`, um Ausnahmen auszulösen, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, obwohl kein Segment geöffnet ist.

  Zulässige Werte

  - `RUNTIME_ERROR`— Löst eine Laufzeitausnahme aus.
  - `LOG_ERROR`— Fehler protokollieren und fortfahren (Standard).
  - `IGNORE_ERROR`— Fehler ignorieren und fortfahren.

  Fehler im Zusammenhang mit fehlenden Segmenten oder Untersegmenten können auftreten, wenn Sie versuchen, einen instrumentierten Client in Startcode zu verwenden, der ausgeführt wird, wenn keine Anfrage geöffnet ist, oder in Code, der einen neuen Thread erzeugt.
- `AWS_XRAY_TRACING_NAME`— Legen Sie den Dienstnamen fest, den das SDK für Segmente verwendet.
- `AWS_XRAY_DAEMON_ADDRESS`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest. Standardmäßig sendet das SDK Trace-Daten an. `127.0.0.1:2000` Verwenden Sie diese Variable, wenn Sie den Daemon so konfiguriert haben, dass er auf einem anderen Port lauscht, oder wenn er auf einem anderen Host läuft.
- `AWS_XRAY_CONTEXT_MISSING`— Legen Sie den Wert fest, um zu bestimmen, wie das SDK mit Fehlern im Zusammenhang mit fehlenden Kontexten umgeht. Fehler aufgrund fehlender Segmente oder Untersegmente können auftreten, wenn Sie versuchen, einen instrumentierten Client in Startup-Code zu verwenden, wenn keine Anfrage geöffnet ist, oder in Code, der einen neuen Thread aufruft.
  - `RUNTIME_ERROR`— Standardmäßig ist das SDK so eingestellt, dass es eine Laufzeitausnahme auslöst.

- `LOG_ERROR`— Ist so eingestellt, dass ein Fehler protokolliert und fortgefahren wird.

Umgebungsvariablen überschreiben äquivalente Werte im Code.

Verwenden von „Configure“ (konfigurieren)

Sie können das X-Ray SDK for Go auch mit `Configure` dieser Methode konfigurieren. `Configure`akzeptiert ein Argument, ein `Config` Objekt, mit den folgenden optionalen Feldern.

DaemonAddr

Diese Zeichenfolge gibt den Host und den Port des X-Ray-Daemon-Listeners an. Falls nicht angegeben, verwendet X-Ray den Wert der `AWS_XRAY_DAEMON_ADDRESS` Umgebungsvariablen. Wenn dieser Wert nicht festgelegt ist, wird "127.0.0.1:2000" verwendet.

ServiceVersion

Diese Zeichenfolge legt die Service-Version fest. Falls nicht angegeben, verwendet X-Ray die leere Zeichenfolge („“).

SamplingStrategy

Das `SamplingStrategy`-Objekt legt fest, welche Ihrer Anwendungsaufrufe verfolgt werden. Falls nicht angegeben, verwendet X-Ray a`LocalizedSamplingStrategy`, was die Strategie wie unter definiert verwendet`xray/resources/DefaultSamplingRules.json`.

StreamingStrategy

Dieses `StreamingStrategy` Objekt gibt an, ob ein Segment gestreamt werden soll, wenn der RequiresStreamingWert true zurückgegeben wird. Falls nicht angegeben, verwendet X-Ray eine, `DefaultStreamingStrategy` die ein abgetastetes Segment streamt, wenn die Anzahl der Untersegmente größer als 20 ist.

ExceptionFormattingStrategy

Dieses `ExceptionFormattingStrategy`-Objekt legt fest, auf welche Weise verschiedene Ausnahmen gehandhabt werden sollen. Falls nicht angegeben, verwendet X-Ray a `DefaultExceptionFormattingStrategy` mit einem `XrayError` of-Type`error`, die Fehlermeldung und Stack-Trace.

Instrumentierung eingehender HTTP-Anfragen mit dem X-Ray SDK for Go

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können das X-Ray SDK verwenden, um eingehende HTTP-Anfragen zu verfolgen, die Ihre Anwendung auf einer EC2 Instance in Amazon EC2 oder Amazon ECS bearbeitet. AWS Elastic Beanstalk

Verwenden Sie `xray.Handler`, um eingehende HTTP-Anforderungen zu instrumentieren. Das X-Ray-SDK SDK for Go implementiert die `http.Handler` Standard-Go-Bibliotheksschnittstelle in der `xray.Handler` Klasse, um Webanfragen abzufangen. Die `xray.Handler`-Klasse umschließt den bereitgestellten `http.Handler` mit `xray.Capture`, wobei der Anfragekontext genutzt, eingehende Header analysiert und bei Bedarf Antwort-Header hinzugefügt werden, und legt HTTP-spezifische Ablaufverfolgungsfelder fest.

Wenn Sie diese Klasse verwenden, um HTTP-Anfragen und -Antworten zu verarbeiten, erstellt das X-Ray SDK for Go ein Segment für jede gesampelte Anfrage. Dieses Segment umfasst Dauer, Methode und Status der HTTP-Anforderung. Die zusätzliche Instrumentierung schafft Untersegmente zu diesem Segment.

> **Note**
>
> Für AWS Lambda Funktionen erstellt Lambda für jede abgetastete Anfrage ein Segment. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Das folgende Beispiel fängt Anfragen auf Port 8000 ab und gibt „Hello!“ zurück als Antwort. Es erzeugt das Segment `myApp` und instrumentiert Aufrufe über jede beliebige Anwendung.

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)
}
```

Jedes Segment hat einen Namen, der Ihre Anwendung in der Service Map identifiziert. Das Segment kann statisch benannt werden, oder Sie können das SDK so konfigurieren, dass es dynamisch auf der Grundlage des Host-Headers in der eingehenden Anfrage benannt wird. Mit der dynamischen Benennung können Sie Traces auf der Grundlage des Domainnamens in der Anfrage gruppieren und einen Standardnamen anwenden, wenn der Name nicht einem erwarteten Muster entspricht (z. B. wenn der Host-Header gefälscht ist).

> **Weitergeleitete Anfragen**
>
> Wenn ein Load Balancer oder ein anderer Vermittler eine Anfrage an Ihre Anwendung weiterleitet, nimmt X-Ray die Client-IP aus dem `X-Forwarded-For` Header in der Anfrage und nicht aus der Quell-IP im IP-Paket. Die Client-IP, die für eine weitergeleitete Anfrage aufgezeichnet wird, kann gefälscht sein und sollte daher nicht als vertrauenswürdig eingestuft werden.

Wenn eine Anfrage weitergeleitet wird, legt das SDK ein zusätzliches Feld im Segment fest, um dies anzuzeigen. Wenn das Segment das Feld enthält, das auf `x_forwarded_for` gesetzt ist`true`, wurde die Client-IP aus dem `X-Forwarded-For` Header in der HTTP-Anfrage übernommen.

Der Handler erzeugt für jede eingehende Anfrage ein Segment mit einem `http`-Block mit folgenden Informationen:

- HTTP-Methode — GET, POST, PUT, DELETE usw.
- Client-Adresse — Die IP-Adresse des Clients, der die Anfrage gesendet hat.
- Antwortcode — Der HTTP-Antwortcode für die abgeschlossene Anfrage.
- Timing — Die Startzeit (als die Anfrage empfangen wurde) und die Endzeit (als die Antwort gesendet wurde).

- Benutzeragent — Der `user-agent` aus der Anfrage.
- Länge des Inhalts — Die Länge `content-length` der Antwort.

Konfiguration einer Segmentbenennungsstrategie

AWS X-Ray verwendet einen Dienstnamen, um Ihre Anwendung zu identifizieren und sie von den anderen Anwendungen, Datenbanken, externen AWS Ressourcen und Ressourcen zu unterscheiden APIs, die Ihre Anwendung verwendet. Wenn das X-Ray SDK Segmente für eingehende Anfragen generiert, zeichnet es den Dienstnamen Ihrer Anwendung im Namensfeld des Segments auf.

Das X-Ray SDK kann Segmente nach dem Hostnamen im HTTP-Anforderungsheader benennen. Dieser Header kann jedoch gefälscht sein, was zu unerwarteten Knoten in Ihrer Service Map führen kann. Um zu verhindern, dass das SDK Segmente aufgrund von Anfragen mit gefälschten Host-Headern falsch benennt, müssen Sie einen Standardnamen für eingehende Anfragen angeben.

Wenn Ihre Anwendung Anfragen für mehrere Domänen bearbeitet, können Sie das SDK so konfigurieren, dass es eine dynamische Benennungsstrategie verwendet, um dies in Segmentnamen widerzuspiegeln. Eine dynamische Benennungsstrategie ermöglicht es dem SDK, den Hostnamen für Anfragen zu verwenden, die einem erwarteten Muster entsprechen, und den Standardnamen auf Anfragen anzuwenden, bei denen dies nicht der Fall ist.

Beispielsweise können Sie über eine einzige Anwendung verfügen, die Anfragen an drei Subdomänen — `www.example.com`api.example.com, und — bedient. `static.example.com` Sie können eine dynamische Benennungsstrategie mit dem Muster verwenden`*.example.com`, um Segmente für jede Subdomain mit einem anderen Namen zu identifizieren, was zu drei Dienstknoten auf der Service-Map führt. Wenn Ihre Anwendung Anfragen mit einem Hostnamen empfängt, der nicht dem Muster entspricht, wird auf der Service Map ein vierter Knoten mit einem von Ihnen angegebenen Fallback-Namen angezeigt.

Wenn Sie denselben Namen für alle Anfragesegmente verwenden möchten, geben Sie bei der Erstellung des Handlers den Namen Ihrer Anwendung ein, wie im vorherigen Abschnitt gezeigt.

> **Note**
>
> Sie können den mit der `AWS_XRAY_TRACING_NAME`-Umgebungsvariablen in Code definierten standardmäßigen Dienstnamen überschreiben.

Eine dynamische Benennungsstrategie definiert ein Muster, dem Hostnamen entsprechen sollten, sowie einen Standardnamen, der verwendet wird, wenn der Hostname in der HTTP-Anforderung nicht mit diesem Muster übereinstimmt. Um Segmente dynamisch zu benennen, verwenden Sie `NewDynamicSegmentNamer` zur Konfiguration des Standardnamens und Musters zum Abgleich.

Example main.go

Wenn der Hostname in der Anforderung dem Muster `*.example.com` entspricht, verwenden Sie den Hostnamen. Verwenden Sie andernfalls `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)
}
```

Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for Go

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Aufrufe tätigt, um Daten AWS-Services zu speichern, in eine Warteschleife zu schreiben oder Benachrichtigungen zu senden, verfolgt das X-Ray SDK for Go die Aufrufe im Downstream in Untersegmenten. Verfolgte Ressourcen AWS-Services und Ressourcen, auf die Sie innerhalb dieser Services zugreifen (z. B. ein Amazon S3 S3-Bucket oder eine Amazon SQS SQS-Warteschlange), werden als Downstream-Knoten auf der Trace-Map in der X-Ray-Konsole angezeigt.

Um AWS SDK-Clients zu verfolgen, umschließen Sie das Client-Objekt mit dem `xray.AWS()`-Aufruf, wie im folgenden Beispiel dargestellt.

Example main.go

```
var dynamo *dynamodb.DynamoDB
func main() {
  dynamo = dynamodb.New(session.Must(session.NewSession()))
  xray.AWS(dynamo.Client)
}
```

Wenn Sie dann den AWS SDK-Client nutzen, verwenden Sie die Version `withContext` der Aufrufmethode und übergeben Sie den `context` aus dem `http.Request`-Objekt, das an den Handler übergeben wurde.

Example main.go — SDK-Aufruf AWS

```
func listTablesWithContext(ctx context.Context) {
  output := dynamo.ListTablesWithContext(ctx, &dynamodb.ListTablesInput{})
  doSomething(output)
}
```

Für alle Dienste können Sie den Namen der aufgerufenen API in der X-Ray-Konsole sehen. Für eine Untergruppe von Diensten fügt das X-Ray SDK dem Segment Informationen hinzu, um die Service Map detaillierter zu gestalten.

Wenn Sie beispielsweise einen Aufruf mit einem instrumentierten DynamoDB-Client tätigen, fügt das SDK den Tabellennamen dem Segment für Aufrufe hinzu, die auf eine Tabelle abzielen. In der Konsole wird jede Tabelle als separater Knoten in der Service Map angezeigt, mit einem generischen DynamoDB-Knoten für Aufrufe, die nicht auf eine Tabelle abzielen.

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements

```
{
  "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",
  }
}
```

Wenn Sie auf benannte Ressourcen zugreifen, werden durch Aufrufe der folgenden Services weitere Knoten in der Service-Übersicht erstellt. Durch Aufrufe, die keinen bestimmten Ressourcen gelten, wird ein generischer Knoten für den Service erstellt.

- Amazon DynamoDB — Tabellenname
- Amazon Simple Storage Service — Bucket und Schlüsselname
- Amazon Simple Queue Service — Name der Warteschlange

Rückverfolgung von Aufrufen an Downstream-HTTP-Webservices mit dem X-Ray SDK for Go

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Microservices oder öffentliches HTTP aufruft, können Sie die verwenden APIs, `xray.Client` um diese Aufrufe als Untersegmente Ihrer Go-Anwendung zu instrumentieren, wie im folgenden Beispiel gezeigt, wobei http-client ein HTTP-Client ist.

Der Client erstellt eine flache Kopie des bereitgestellten HTTP-Clients. Standardmäßig ist Roundtripper mit eingeschlossen. `http.DefaultClient` `xray.RoundTripper`

Example

<caption>main.go — HTTP-Client</caption>

```
myClient := xray.Client(http-client)
```

<caption>main.go — Verfolgen Sie den Downstream-HTTP-Aufruf mit der ctxhttp-Bibliothek</caption>

Das folgende Beispiel instrumentiert den ausgehenden HTTP-Aufruf mit der ctxhttp-Bibliothek unter Verwendung von. `xray.Client` `ctx`kann vom Upstream-Aufruf aus übergeben werden. Dadurch wird sichergestellt, dass der bestehende Segmentkontext verwendet wird. X-Ray erlaubt beispielsweise nicht, dass ein neues Segment innerhalb einer Lambda-Funktion erstellt wird, sodass der bestehende Lambda-Segmentkontext verwendet werden sollte.

```
resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

Verfolgen von SQL-Abfragen mit dem X-Ray SDK for Go

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

So können Sie SQL-Aufrufe von PostgreSQL oder MySQL rückverfolgen, die `sql.Open`-Aufrufe von `xray.SQLContext` ersetzen, wie im folgenden Beispiel dargestellt. Verwenden Sie nach URLs Möglichkeit anstelle von Konfigurationszeichenfolgen.

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
```

```
}
```

Generieren benutzerdefinierter Untersegmente mit dem X-Ray SDK for Go

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

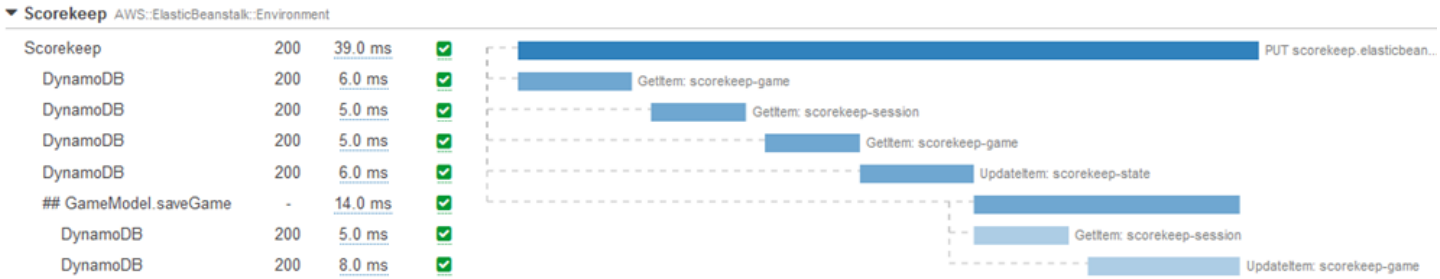
Untersegmente erweitern das Segment eines Traces um Details über die Arbeit, die zur Bearbeitung einer Anfrage geleistet wurde. Jedes Mal, wenn Sie einen Anruf mit einem instrumentierten Client tätigen, zeichnet das X-Ray-SDK die in einem Untersegment generierten Informationen auf. Sie können zusätzliche Untersegmente erstellen, um andere Untersegmente zu gruppieren, die Leistung eines Codeabschnitts zu messen oder Anmerkungen und Metadaten aufzuzeichnen.

Mit der `Capture`-Methode legen Sie ein Untersegment um eine Funktion herum an.

Example main.go — Benutzerdefiniertes Untersegment

```
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)
  })
```

Der folgende Screenshot zeigt ein Beispiel dafür, wie das `saveGame`-Untersegment in Ablaufverfolgungen für die Anwendung `Scorekeep` aussehen kann.



Fügen Sie mit dem X-Ray SDK for Go Anmerkungen und Metadaten zu Segmenten hinzu

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können zusätzliche Informationen über Anfragen, die Umgebung oder Ihre Anwendung mit Anmerkungen und Metadaten aufzeichnen. Sie können Anmerkungen und Metadaten zu den Segmenten hinzufügen, die das X-Ray SDK erstellt, oder zu benutzerdefinierten Untersegmenten, die Sie erstellen.

Anmerkungen sind Schlüssel-Wert-Paare mit Zeichenfolgen-, Zahlen- oder booleschen Werten. Anmerkungen sind für die Verwendung mit Filterausdrücken indexiert. Berücksichtigen Sie Anmerkungen, um Daten zur Gruppierung von Ablaufverfolgungen in der Konsole zu verwenden, oder wenn Sie die `GetTraceSummaries`-API aufrufen.

Metadaten sind Schlüssel-Wert-Paare, die Werte beliebigen Typs enthalten können, einschließlich Objekte und Listen, aber nicht für die Verwendung mit Filterausdrücken indexiert sind. Verwenden Sie Metadaten, um zusätzliche Daten aufzuzeichnen, die Sie im Trace speichern möchten, aber nicht für die Suche verwenden müssen.

Zusätzlich zu Anmerkungen und Metadaten können Sie auch Benutzer-ID-Zeichenfolgen in Segmenten aufzeichnen. Benutzer IDs werden in einem separaten Feld in Segmenten aufgezeichnet und für die Verwendung bei der Suche indexiert.

Sections

Anmerkungen mit dem X-Ray SDK for Go aufnehmen

Verwenden Sie Anmerkungen, um Informationen zu Segmenten, die zur Suche indiziert werden sollten, aufzuzeichnen.

Anmerkung zu Anforderungen

- Schlüssel — Der Schlüssel für eine X-Ray-Anmerkung kann bis zu 500 alphanumerische Zeichen enthalten. Sie können keine anderen Leerzeichen oder Symbole als einen Punkt oder Punkt (.) verwenden
- Werte — Der Wert für eine X-Ray-Anmerkung kann bis zu 1.000 Unicode-Zeichen enthalten.
- Die Anzahl der Anmerkungen — Sie können bis zu 50 Anmerkungen pro Spur verwenden.

Um Anmerkungen aufzuzeichnen, rufen Sie `AddAnnotation` mit einer Zeichenfolge auf, die die Metadaten enthält, die Sie dem Segment zuordnen möchten.

```
xray.AddAnnotation(key string, value interface{})
```

Das SDK zeichnet Anmerkungen als Schlüssel-Wert-Paare in einem `annotations`-Objekt im Segmentdokument auf. Wenn `AddAnnotation` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im selben Segment überschrieben.

Nutzen Sie das `annotation[key]`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen durch Anmerkungen mit bestimmten Werten zu finden.

Metadaten mit dem X-Ray SDK for Go aufnehmen

Verwenden Sie Metadaten, um Segmentinformationen aufzuzeichnen, die nicht zur Suche indiziert werden müssen.

Um Metadaten aufzuzeichnen, rufen Sie `AddMetadata` mit einer Zeichenfolge auf, die die Metadaten enthält, die Sie dem Segment zuordnen möchten.

```
xray.AddMetadata(key string, value interface{})
```

Benutzer IDs mit dem X-Ray SDK for Go aufzeichnen

Zeichnen Sie Segmente von Benutzern IDs auf Anfrage auf, um den Benutzer zu identifizieren, der die Anfrage gesendet hat.

Um den Benutzer aufzuzeichnen IDs

1. Eine Referenz des aktuellen Segments finden Sie unter `AWSXRay`.

   ```
   import (
     "context"
     "github.com/aws/aws-xray-sdk-go/xray"
   )

   mySegment := xray.GetSegment(context)
   ```

2. Rufen Sie `setUser` mit einer Zeichenfolgen-ID des Benutzers auf, der die Anforderung gesendet hat.

   ```
   mySegment.User = "U12345"
   ```

Nutzen Sie das `user`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen einer Benutzer-ID zu finden.

Arbeiten mit Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Es gibt zwei Möglichkeiten, Ihre Java-Anwendung so zu instrumentieren, dass sie Traces an X-Ray sendet:

- AWS Distro for OpenTelemetry Java — Eine AWS Distribution, die eine Reihe von Open-Source-Bibliotheken zum Senden korrelierter Metriken und Traces über AWS Distro for Collector an mehrere AWS Überwachungslösungen CloudWatch AWS X-Ray, darunter Amazon und Amazon OpenSearch Service, bereitstellt. OpenTelemetry
- AWS X-Ray SDK for Java — Eine Reihe von Bibliotheken zum Generieren und Senden von Traces an X-Ray über den X-Ray-Daemon.

Weitere Informationen finden Sie unter Wahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs.

AWS Distro für Java OpenTelemetry

Mit der AWS Distribution für OpenTelemetry (ADOT) Java können Sie Ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Monitoring-Lösungen wie Amazon und Amazon CloudWatch Service AWS X-Ray senden. OpenSearch Die Verwendung von X-Ray mit ADOT erfordert zwei Komponenten: ein OpenTelemetry SDK, das für die Verwendung mit X-Ray aktiviert ist, und das AWS Distro for OpenTelemetry Collector, das für die Verwendung mit X-Ray aktiviert ist. ADOT Java bietet Unterstützung für automatische Instrumentierung, sodass Ihre Anwendung Traces ohne Codeänderungen senden kann.

Informationen zu den ersten Schritten finden Sie in der Dokumentation zu AWS Distro for OpenTelemetry Java.

Weitere Informationen zur Verwendung von AWS Distro for OpenTelemetry with AWS X-Ray und anderen AWS-Services finden Sie unter AWS Distro for OpenTelemetry oder Distro for Documentation AWS . OpenTelemetry

Weitere Informationen zur Sprachunterstützung und -verwendung finden Sie unter AWS Observability on. GitHub

AWS X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray SDK for Java besteht aus einer Reihe von Bibliotheken für Java-Webanwendungen, die Klassen und Methoden zum Generieren und Senden von Trace-Daten an den X-Ray-Daemon bereitstellen. Zu den Trace-Daten gehören Informationen über eingehende HTTP-Anfragen, die von der Anwendung bedient werden, sowie über Aufrufe, die die Anwendung mithilfe des AWS SDK, HTTP-Clients oder eines SQL-Datenbank-Connectors an Downstream-Dienste sendet. Sie können Segmente auch manuell erstellen und Debug-Informationen Anmerkungen und Metadaten hinzufügen.

Das X-Ray SDK for Java ist ein Open-Source-Projekt. Du kannst das Projekt verfolgen und Issues und Pull-Requests einreichen auf GitHub: github. com/aws/aws- xray-sdk-java

Fügen Sie zunächst `AWSXRayServletFilter` als Servlet-Filter hinzu, um eingehende Anforderungen zu verfolgen. Ein Servlet-Filter erstellt ein Segment. Während das Segment geöffnet ist, können Sie die SDK-Client-Methoden nutzen, um dem Segment Informationen hinzuzufügen, Untersegmente zu erstellen und nachgelagerte Aufrufe rückzuverfolgen. Das SDK erfasst auch automatisch Ausnahmen, die Ihre Anwendung ausgibt, während das Segment geöffnet ist.

Ab Version 1.3 können Sie Ihre Anwendung mit aspekt-orientierter Programmierung (AOP) in Spring instrumentieren. Das bedeutet, dass Sie Ihre Anwendung instrumentieren können, während sie läuft AWS, ohne der Laufzeit Ihrer Anwendung Code hinzufügen zu müssen.

Verwenden Sie als Nächstes das X-Ray-SDK SDK for Java, um Ihre AWS SDK für Java Clients zu instrumentieren, indem Sie das SDK Instrumentor-Untermodul in Ihre Build-Konfiguration aufnehmen. Immer wenn Sie mit einem instrumentierten Client einen Downstream AWS-Service oder eine Ressource aufrufen, zeichnet das SDK Informationen über den Anruf in einem Untersegment auf. AWS-Services und die Ressourcen, auf die Sie innerhalb der Services zugreifen, werden in der Trace-Map als Downstream-Knoten angezeigt, sodass Sie Fehler und Drosselungsprobleme bei einzelnen Verbindungen leichter identifizieren können.

Wenn Sie nicht alle Downstream-Aufrufe instrumentieren möchten AWS-Services, können Sie das Instrumentor-Untermodul weglassen und auswählen, welche Clients instrumentiert werden sollen. Instrumentieren Sie einzelne Clients, indem Sie einem `TracingHandler` AWS SDK-Serviceclient einen hinzufügen.

Andere Submodule des X-Ray SDK for Java bieten Instrumentierung für Downstream-Aufrufe von HTTP-Web APIs - und SQL-Datenbanken. Sie können das X-Ray-SDK SDK for Java Java-Versionen von `HTTPClient` und `HTTPClientBuilder` im Apache HTTP-Submodul verwenden, um Apache HTTP-Clients zu instrumentieren. Um SQL-Anfragen zu instrumentieren, fügen Sie der Datenquelle den SDK-Interceptor hinzu.

Nachdem Sie das SDK verwendet haben, passen Sie sein Verhalten an, indem Sie den Rekorder und den Servlet-Filter konfigurieren. Sie können Plugins zum Festhalten von Daten über die Datenverarbeitungsressourcen, auf denen Ihre Anwendung ausgeführt wird, hinzufügen, das Samplingverhalten durch Samplingregeln anpassen und Protokollebenen einrichten, um mehr oder weniger Informationen von dem SDK in Ihren Anwendungsprotokollen zu sehen.

Zeichnen Sie zusätzliche Informationen zu Anforderungen und den Aufgaben, die Ihre Anwendung ausführt, in Anmerkungen und Metadaten auf. Anmerkungen sind einfache Schlüsselwertpaare, die für die Verwendung mit Filterausdrücken indiziert werden, damit Sie nach Ablaufverfolgen mit bestimmten Daten suchen können. Metadateneinträge sind weniger einschränkend und können ganze Objekte und Arrays aufzeichnen – alle Daten, die in eine JSON zusammengefasst werden können.

> **Anmerkungen und Metadaten**
>
> Anmerkungen und Metadaten sind beliebiger Text, den Sie Segmenten mit dem X-Ray SDK hinzufügen. Anmerkungen werden für die Verwendung mit Filterausdrücken indexiert. Metadaten werden nicht indexiert, können aber im Rohsegment mit der X-Ray-Konsole oder API angezeigt werden. Jeder, dem Sie Lesezugriff auf X-Ray gewähren, kann diese Daten einsehen.

Wenn Sie viele instrumentierte Clients in Ihrem Code haben, kann ein einzelnes Anfragesegment viele Untersegmente enthalten, eines für jeden Aufruf mit einem instrumentierten Client. Sie können Untersegmente organisieren und gruppieren, indem Sie Client-Aufrufe in benutzerdefinierten Untersegmenten zusammenfassen. Sie können ein benutzerdefiniertes Untersegment für eine ganze Funktion oder eine Code-Abschnitt erstellen und Metadaten und Anmerkungen im Untersegment festhalten, anstatt alles im übergeordneten Segment aufzuzeichnen.

Untermodule

Sie können das X-Ray SDK for Java von Maven herunterladen. Das X-Ray-SDK SDK for Java ist je nach Anwendungsfall in Untermodule aufgeteilt und enthält eine Materialliste für die Versionsverwaltung:

- `aws-xray-recorder-sdk-core`(erforderlich) — Grundlegende Funktionen für die Erstellung von Segmenten und die Übertragung von Segmenten. Umfasst `AWSXRayServletFilter` für die Instrumentierung eingehender Anforderungen.
- `aws-xray-recorder-sdk-aws-sdk`— Instrumentiert Aufrufe, die mit AWS SDK für Java Clients AWS-Services getätigt wurden, indem ein Tracing-Client als Request-Handler hinzugefügt wird.
- `aws-xray-recorder-sdk-aws-sdk-v2`— Instrumentiert Aufrufe, die mit Clients der Version AWS SDK für Java 2.2 und höher AWS-Services getätigt wurden, indem ein Tracing-Client als Request-Interzeptor hinzugefügt wird.
- `aws-xray-recorder-sdk-aws-sdk-instrumentor`— Instrumentiert alle `aws-xray-recorder-sdk-aws-sdk` Clients automatisch. AWS SDK für Java
- `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor`— `aws-xray-recorder-sdk-aws-sdk-v2` Instrumentiert alle Clients ab Version AWS SDK für Java 2.2 automatisch.
- `aws-xray-recorder-sdk-apache-http`— Instrumentiert ausgehende HTTP-Aufrufe, die mit Apache HTTP-Clients getätigt wurden.

- `aws-xray-recorder-sdk-spring`— Stellt Interzeptoren für Spring AOP Framework-Anwendungen bereit.
- `aws-xray-recorder-sdk-sql-postgres`— Instrumentiert ausgehende Aufrufe an eine PostgreSQL-Datenbank, die mit JDBC getätigt wurden.
- `aws-xray-recorder-sdk-sql-mysql`— Instrumentiert ausgehende Aufrufe an eine MySQL-Datenbank, die mit JDBC getätigt wurden.
- `aws-xray-recorder-sdk-bom`— Stellt eine Stückliste bereit, anhand derer Sie die Version angeben können, die für alle Submodule verwendet werden soll.
- `aws-xray-recorder-sdk-metrics`— Veröffentlichen Sie CloudWatch Amazon-Metriken ohne Stichproben aus Ihren gesammelten X-Ray-Segmenten.

Wenn Sie Maven oder Gradle verwenden, um Ihre Anwendung zu erstellen, fügen Sie das X-Ray SDK for Java zu Ihrer Build-Konfiguration hinzu.

Eine Referenzdokumentation der Klassen und Methoden des SDK finden Sie unter AWS X-Ray SDK for Java API Reference.

Voraussetzungen

Das X-Ray SDK for Java erfordert Java 8 oder höher, Servlet API 3, das AWS SDK und Jackson.

Das SDK hängt von den folgenden Bibliotheken zur Kompilierungs- und Laufzeit ab:

- AWS SDK for Java Version 1.11.398 oder höher
- Servlet API 3.1.0

Diese Abhängigkeiten werden in der `pom.xml`-Datei des SDK deklariert und automatisch eingefügt, wenn Sie Maven oder Gradle für die Build-Erstellung verwenden.

Wenn Sie eine Bibliothek verwenden, die im X-Ray SDK for Java enthalten ist, müssen Sie die mitgelieferte Version verwenden. Wenn Sie beispielsweise bereits zur Laufzeit von Jackson abhängen und für diese Abhängigkeit JAR-Dateien in Ihre Bereitstellung aufnehmen, müssen Sie diese JAR-Dateien entfernen, da das SDK JAR eigene Versionen von Jackson-Bibliotheken umfasst.

Abhängigkeitsmanagement

Das X-Ray SDK for Java ist bei Maven erhältlich:

- Gruppe — `com.amazonaws`
- Artifact — `aws-xray-recorder-sdk-bom`
- Version – `2.11.0`

Wenn Sie Ihre Anwendung mit Maven erstellen, fügen Sie das SDK als Abhängigkeit in Ihrer `pom.xml`-Datei hinzu.

Example pom.xml - Abhängigkeiten

```
<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>
```

Für Gradle fügen Sie das SDK als Kompilierungszeitabhängigkeit in Ihrer `build.gradle`-Datei hinzu.

Example build.gradle – Abhängigkeiten

```
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')
    }
}
```

Wenn Sie Elastic Beanstalk für die Bereitstellung Ihrer Anwendung verwenden, können Sie Maven oder Gradle verwenden, um bei jeder Bereitstellung eine On-Instance zu erstellen, anstatt ein großes Archiv mit all Ihren Abhängigkeiten zu erstellen und hochzuladen. Für ein Beispiel, bei dem Gradle verwendet wird, sehen Sie sich die Beispielanwendung an.

AWS X-Ray Agent für automatische Instrumentierung für Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan

für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon
Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Der AWS X-Ray Autoinstrumentation-Agent für Java ist eine Tracing-Lösung, die Ihre Java-Webanwendungen mit minimalem Entwicklungsaufwand instrumentiert. Der Agent ermöglicht die Ablaufverfolgung für Servlet-basierte Anwendungen und alle Downstream-Anfragen des Agenten, die mit unterstützten Frameworks und Bibliotheken gestellt wurden. Dazu gehören Downstream-HTTP-Anfragen von Apache, AWS SDK-Anfragen und SQL-Abfragen, die mit einem JDBC-Treiber gestellt wurden. Der Agent verbreitet den X-Ray-Kontext, einschließlich aller aktiven Segmente und Untersegmente, über Threads hinweg. Alle Konfigurationen und die Vielseitigkeit des X-Ray-SDK sind weiterhin mit dem Java-Agenten verfügbar. Es wurden geeignete Standardeinstellungen ausgewählt, um sicherzustellen, dass der Agent mit minimalem Aufwand arbeitet.

Die X-Ray-Agentenlösung eignet sich am besten für Servlet-basierte Java-Webanwendungsserver mit Request-Response. Wenn Ihre Anwendung ein asynchrones Framework verwendet oder nicht gut als Request-Response-Service modelliert ist, sollten Sie stattdessen eine manuelle Instrumentierung mit dem SDK in Betracht ziehen.

Der X-Ray-Agent wird mit dem Distributed Systems Comprehension Toolkit (Di) erstellt. SCo Di SCo ist ein Open-Source-Framework zum Erstellen von Java-Agenten, die in verteilten Systemen verwendet werden können. Es ist zwar nicht notwendig, Di zu verstehen, SCo um den X-Ray-Agenten verwenden zu können, aber Sie können mehr über das Projekt erfahren, indem Sie die Homepage unter besuchen GitHub. Der X-Ray-Agent ist ebenfalls vollständig als Open Source verfügbar. Um den Quellcode einzusehen, Beiträge zu leisten oder Probleme mit dem Agenten zu äußern, besuchen Sie das entsprechende Repository unter GitHub.

Beispielanwendung

Die eb-java-scorekeepProbenapplikation ist so angepasst, dass sie mit dem Röntgenmittel instrumentiert werden kann. Dieser Zweig enthält keine Servlet-Filter- oder Rekorderkonfiguration, da diese Funktionen vom Agenten ausgeführt werden. Um die Anwendung lokal oder mithilfe von AWS Ressourcen auszuführen, folgen Sie den Schritten in der Readme-Datei der Beispielanwendung. Die Anweisungen zur Verwendung der Beispiel-App zum Generieren von Röntgenspuren finden Sie im Tutorial der Beispiel-App.

Erste Schritte

Gehen Sie wie folgt vor, um mit dem X-Ray-Java-Agenten für automatische Instrumentierung in Ihrer eigenen Anwendung zu beginnen.

1. Führen Sie den X-Ray-Daemon in Ihrer Umgebung aus. Weitere Informationen finden Sie unter AWS X-Ray Dämon.
2. Laden Sie die neueste Version des Agenten herunter. Entpacken Sie das Archiv und notieren Sie sich seinen Speicherort in Ihrem Dateisystem. Sein Inhalt sollte wie folgt aussehen.

   ```
   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. Ändern Sie die JVM-Argumente Ihrer Anwendung so, dass sie Folgendes enthalten, wodurch der Agent aktiviert wird. Stellen Sie sicher, dass das `-javaagent` Argument vor dem `-jar` Argument steht, falls zutreffend. Der Prozess zum Ändern von JVM-Argumenten hängt von den Tools und Frameworks ab, die Sie zum Starten Ihres Java-Servers verwenden. Spezifische Anleitungen finden Sie in der Dokumentation Ihres Server-Frameworks.

   ```
   -javaagent:/<path-to-disco>/disco-java-agent.jar=pluginPath=/<path-to-disco>/disco-
   plugins
   ```

4. Um festzulegen, wie der Name Ihrer Anwendung auf der X-Ray-Konsole angezeigt wird, legen Sie die `AWS_XRAY_TRACING_NAME` Umgebungsvariable oder die `com.amazonaws.xray.strategy.tracingName` Systemeigenschaft fest. Wenn kein Name angegeben wird, wird ein Standardname verwendet.
5. Starten Sie Ihren Server oder Container neu. Eingehende Anfragen und ihre Downstream-Aufrufe werden jetzt verfolgt. Wenn Sie nicht die erwarteten Ergebnisse sehen, finden Sie weitere Informationen unterthe section called “Fehlerbehebung”.

Konfiguration

Der X-Ray-Agent wird durch eine externe, vom Benutzer bereitgestellte JSON-Datei konfiguriert. Standardmäßig wird diese Datei im Stammverzeichnis des Klassenpfads des Benutzers

(z. B. in seinem `resources` Verzeichnis) benannt. `xray-agent.json` Sie können einen benutzerdefinierten Speicherort für die Konfigurationsdatei konfigurieren, indem Sie die `com.amazonaws.xray.configFile` Systemeigenschaft auf den absoluten Dateisystempfad Ihrer Konfigurationsdatei setzen.

Als Nächstes wird ein Beispiel für eine Konfigurationsdatei angezeigt.

```
{
    "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
}
```

Konfigurationsspezifikation

In der folgenden Tabelle werden gültige Werte für jede Eigenschaft beschrieben. Bei Eigenschaftsnamen wird zwischen Groß- und Kleinschreibung unterschieden, bei ihren Schlüsseln jedoch nicht. Bei Eigenschaften, die durch Umgebungsvariablen und Systemeigenschaften außer Kraft gesetzt werden können, ist die Prioritätsreihenfolge immer die Umgebungsvariable, dann die Systemeigenschaft und dann die Konfigurationsdatei. Hinweise zu Eigenschaften, die Sie überschreiben können, finden Sie unter. Umgebungsvariablen Alle Felder sind optional.

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
serviceName	Zeichenfolge	Jede Zeichenfolge	Der Name Ihres instrumen	AWS_XRAY_TRACING_NAME	com.amazonaws.xray.strategy	XRayInstrumentedService

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
			tierten Dienstes, so wie er in der X-Ray-Konsole angezeigt wird.		.tracingName	
contextMissingStrategy	Zeichenfolge	LOG_ERROR, IGNORE_ERROR	Die Aktion, die der Agent ausführt, wenn er versucht, den X-Ray-Segmentkontext zu verwenden, aber keiner vorhanden ist.	AWS_XRAY_CONTEXT_MISSING	com.amazonaws.xray.strategy.contextMissingStrategy	LOG_ERROR

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
Daemon-Adresse	Zeichenfolge	Formatierte IP-Adresse und Port oder Liste von TCP- und UDP-Adressen	Die Adresse, die der Agent für die Kommunikation mit dem X-Ray-Daemon verwendet.	AWS_XRAY_DAEMON_ADRESSE	com.amazonaws.xray.emitter.DAEMON-Adresse	127.0.0.1:2000
Ablaufverfolgung aktiviert	Boolesch	Wahr, falsch	Ermöglicht die Instrumentierung durch den X-Ray-Agenten.	AWS_XRAY_TRACING_ENABLED	com.amazonaws.xray.TracingEnabled	TRUE

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
Strategie für die Probenahme	Zeichenfolge	ZENTRAL, LOKAL, KEINER, ALLES	Die vom Agenten verwendete Probenahmestrategie. ALL erfasst alle Anfragen, NONE erfasst keine Anfragen. Siehe Regeln für die Probenahme.	–	–	ZENTRALE
traceIdInjectionPräfix	Zeichenfolge	Jede Zeichenfolge	Schließt das angegebene Präfix vor der Eingabe von Trace IDs in die Protokolle ein.	–	–	Keine (leere Zeichenfolge)

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
samplingRulesManifest	Zeichenfolge	Ein absoluter Dateipfad	Der Pfad zu einer Datei mit benutzerdefinierten Stichprobenregeln, die als Quelle für Stichprobenregeln für die lokale Stichprobenstrategie oder als Ausweichregeln für die zentrale Strategie verwendet werden soll.	–	–	DefaultSamplingRules.json

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
awsServiceHandlerManifest	Zeichenfolge	Ein absoluter Dateipfad	Der Pfad zu einer Zulassungsliste für benutzerdefinierte Parameter, die zusätzliche Informationen von AWS SDK-Clients erfasst.	–	–	DefaultOperationParameterWhitelist.json
awsSdkVersion	Ganzzahl	1, 2	Version des AWS SDK for Java, das Sie verwenden. Wird ignoriert, wenn `awsServiceHandlerManifest` es nicht auch gesetzt ist.	–	–	2

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
maxStackTraceLänge	Ganzzahl	Nichtnegative Ganzzahlen	Die maximale Anzahl von Zeilen eines Stack-Trace, die in einem Trace aufgezeichnet werden sollen.	–	–	50
Streaming Threshold	Ganzzahl	Nichtnegative Ganzzahlen	Nachdem mindestens so viele Untersegmente geschlossen wurden, werden sie an den Daemon gestreamt, um zu verhindern, dass die Blöcke out-of-band zu groß werden.	–	–	100

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
traceIdInjection	Boolesch	Wahr, falsch	Aktiviert die X-Ray-Trace-ID-Injektion in Protokolle, wenn die in der Protokollierungskonfiguration beschriebenen Abhängigkeiten und die Konfiguration ebenfalls hinzugefügt werden. Sonst tut es nichts.	–	–	TRUE

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
Plugins aktiviert	Boolesch	Wahr, falsch	Aktiviert Plugins, die Metadaten über die AWS Umgebungen aufzeichnen, in denen Sie arbeiten. Siehe Plugins.	–	–	TRUE
collectSqlQueries	Boolesch	Wahr, Falsch	Zeichnet SQL-Abfragezeichenfolgen nach bestem Wissen und Gewissen in SQL-Untersegmenten auf.	–	–	FALSE

Name der Eigenschaft	Typ	Zulässige Werte	Description	Umgebungsvariable	Systemeigenschaft	Standard
Kontextweiterleitung	Boolesch	Wahr, falsch	Gibt automatisch den X-Ray-Kontext zwischen Threads weiter, falls der Wert wahr ist. Andernfalls verwendet Thread Local, um den Kontext zu speichern, und eine manuelle Weitergabe zwischen Threads ist erforderlich.	–	–	TRUE

Protokollierungskonfiguration

Das Log Level des X-Ray-Agenten kann auf die gleiche Weise wie das X-Ray SDK for Java konfiguriert werden. Weitere Informationen Protokollierung zur Konfiguration der Protokollierung mit dem X-Ray SDK for Java finden Sie unter.

Manuelle Instrumentierung

Wenn Sie zusätzlich zur automatischen Instrumentierung des Agenten eine manuelle Instrumentierung durchführen möchten, fügen Sie das X-Ray SDK als Abhängigkeit zu Ihrem Projekt hinzu. Beachten Sie, dass die unter Tracing Incoming Requests genannten benutzerdefinierten Servlet-Filter des SDK nicht mit dem X-Ray-Agenten kompatibel sind.

> **Note**
>
> Sie müssen die neueste Version des X-Ray-SDK verwenden, um die manuelle Instrumentierung durchzuführen und gleichzeitig den Agenten zu verwenden.

Wenn Sie in einem Maven-Projekt arbeiten, fügen Sie Ihrer `pom.xml` Datei die folgenden Abhängigkeiten hinzu.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
    <version>2.11.0</version>
  </dependency>
  </dependencies>
```

Wenn Sie in einem Gradle-Projekt arbeiten, fügen Sie Ihrer `build.gradle` Datei die folgenden Abhängigkeiten hinzu.

```
implementation 'com.amazonaws:aws-xray-recorder-sdk-core:2.11.0'
```

Sie können während der Verwendung des Agenten zusätzlich zu Anmerkungen, Metadaten und Benutzern benutzerdefinierte Untersegmente hinzufügen, genau wie IDs beim normalen SDK. Der Agent verbreitet den Kontext automatisch zwischen Threads, sodass bei der Arbeit mit Multithread-Anwendungen keine Behelfslösungen für die Übertragung des Kontexts erforderlich sein sollten.

Fehlerbehebung

Da der Agent über eine vollautomatische Instrumentierung verfügt, kann es schwierig sein, die Ursache eines Problems zu ermitteln, wenn Probleme auftreten. Wenn der X-Ray-Agent bei Ihnen nicht wie erwartet funktioniert, überprüfen Sie die folgenden Probleme und Lösungen. Der X-Ray-Agent und das SDK verwenden Jakarta Commons Logging (JCL). Um die Logging-Ausgabe zu

sehen, stellen Sie sicher, dass sich eine Bridge, die JCL mit Ihrem Logging-Backend verbindet, im Klassenpfad befindet, wie im folgenden Beispiel: oder. `log4j-jcl` `jcl-over-slf4j`

Problem: Ich habe den Java-Agenten in meiner Anwendung aktiviert, sehe aber nichts auf der X-Ray-Konsole

Läuft der X-Ray-Daemon auf demselben Computer?

Falls nicht, lesen Sie in der Dokumentation zum X-Ray-Daemon nach, um ihn einzurichten.

Sehen Sie in Ihren Anwendungsprotokollen eine Meldung wie „Initialisierung des X-Ray Agent Recorders“?

Wenn Sie den Agenten korrekt zu Ihrer Anwendung hinzugefügt haben, wird diese Meldung auf der INFO-Ebene protokolliert, wenn Ihre Anwendung gestartet wird, bevor sie Anfragen entgegennimmt. Wenn diese Meldung nicht angezeigt wird, läuft der Java-Agent nicht zusammen mit Ihrem Java-Prozess. Stellen Sie sicher, dass Sie alle Einrichtungsschritte korrekt und ohne Tippfehler ausgeführt haben.

Sehen Sie in Ihren Anwendungsprotokollen mehrere Fehlermeldungen mit der Aufschrift „Ausnahme fehlt AWS X-Ray im Kontext unterdrücken“?

Diese Fehler treten auf, weil der Agent versucht, Downstream-Anfragen wie AWS SDK-Anfragen oder SQL-Abfragen zu instrumentieren, der Agent aber nicht in der Lage war, automatisch ein Segment zu erstellen. Wenn Sie viele dieser Fehler sehen, ist der Agent möglicherweise nicht das beste Tool für Ihren Anwendungsfall und Sie sollten stattdessen eine manuelle Instrumentierung mit dem X-Ray-SDK in Betracht ziehen. Alternativ können Sie die X-Ray SDK-Debug-Logs aktivieren, um den Stack-Trace zu sehen, wo die kontextlosen Ausnahmen auftreten. Sie können diese Teile Ihres Codes mit benutzerdefinierten Segmenten umschließen, wodurch diese Fehler behoben werden sollten. Ein Beispiel für das Umschließen von Downstream-Anfragen mit benutzerdefinierten Segmenten finden Sie im Beispielcode unter Instrumentieren von Startcode.

Problem: Einige der Segmente, die ich erwarte, erscheinen nicht auf der X-Ray-Konsole

Verwendet Ihre Anwendung Multithreading?

Wenn einige Segmente, von denen Sie erwarten, dass sie erstellt werden, nicht in Ihrer Konsole erscheinen, könnte dies an Hintergrund-Threads in Ihrer Anwendung liegen. Wenn Ihre Anwendung Aufgaben mithilfe von Hintergrund-Threads ausführt, die „Fire and Forget“ sind, wie z. B. einen einmaligen Aufruf einer Lambda-Funktion mit dem AWS SDK oder das regelmäßige Abfragen einiger HTTP-Endpunkte, kann dies den Agenten verwirren, während er den Kontext zwischen Threads

verbreitet. Um zu überprüfen, ob dies Ihr Problem ist, aktivieren Sie die X-Ray SDK-Debug-Logs und suchen Sie nach Meldungen wie: Segment wird nicht ausgegeben, da es Untersegmente in Bearbeitung übergeordnet <NAME >hat. Um dieses Problem zu umgehen, können Sie versuchen, den Hintergrund-Threads beizutreten, bevor Ihr Server zurückkehrt, um sicherzustellen, dass die gesamte in ihnen geleistete Arbeit aufgezeichnet wird. Oder Sie können die `contextPropagation` Konfiguration des Agenten so einstellen, dass die `false` Kontextweiterleitung in Hintergrund-Threads deaktiviert wird. In diesem Fall müssen Sie diese Threads manuell mit benutzerdefinierten Segmenten instrumentieren oder die Ausnahmen ignorieren, die sie erzeugen, wenn der Kontext fehlt.

Haben Sie Stichprobenregeln eingerichtet?

Wenn scheinbar zufällige oder unerwartete Segmente auf der X-Ray-Konsole erscheinen oder die Segmente, von denen Sie erwarten, dass sie sich auf der Konsole befinden, nicht, liegt möglicherweise ein Sampling-Problem vor. Der X-Ray-Agent wendet zentralisierte Stichproben auf alle von ihm erstellten Segmente an und verwendet dabei die Regeln der X-Ray-Konsole. Die Standardregel lautet, dass 1 Segment pro Sekunde plus 5% der nachfolgenden Segmente abgetastet werden. Das bedeutet, dass Segmente, die schnell mit dem Agenten erstellt werden, möglicherweise nicht gesampelt werden. Um dieses Problem zu lösen, sollten Sie auf der X-Ray-Konsole benutzerdefinierte Sampling-Regeln erstellen, die die gewünschten Segmente entsprechend abtasten. Weitere Informationen finden Sie unter Sampling.

Konfiguration des X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK SDK for Java enthält eine Klasse mit dem Namen`AWSXRay`, die den globalen Rekorder bereitstellt. Dies ist ein `TracingHandler`, mit dem Sie Ihren Code instrumentieren

können. Sie können die globale Aufzeichnung so konfigurieren, dass der `AWSXRayServletFilter`, der Segmente für eingehende HTTP-Aufrufe erstellt, angepasst wird.

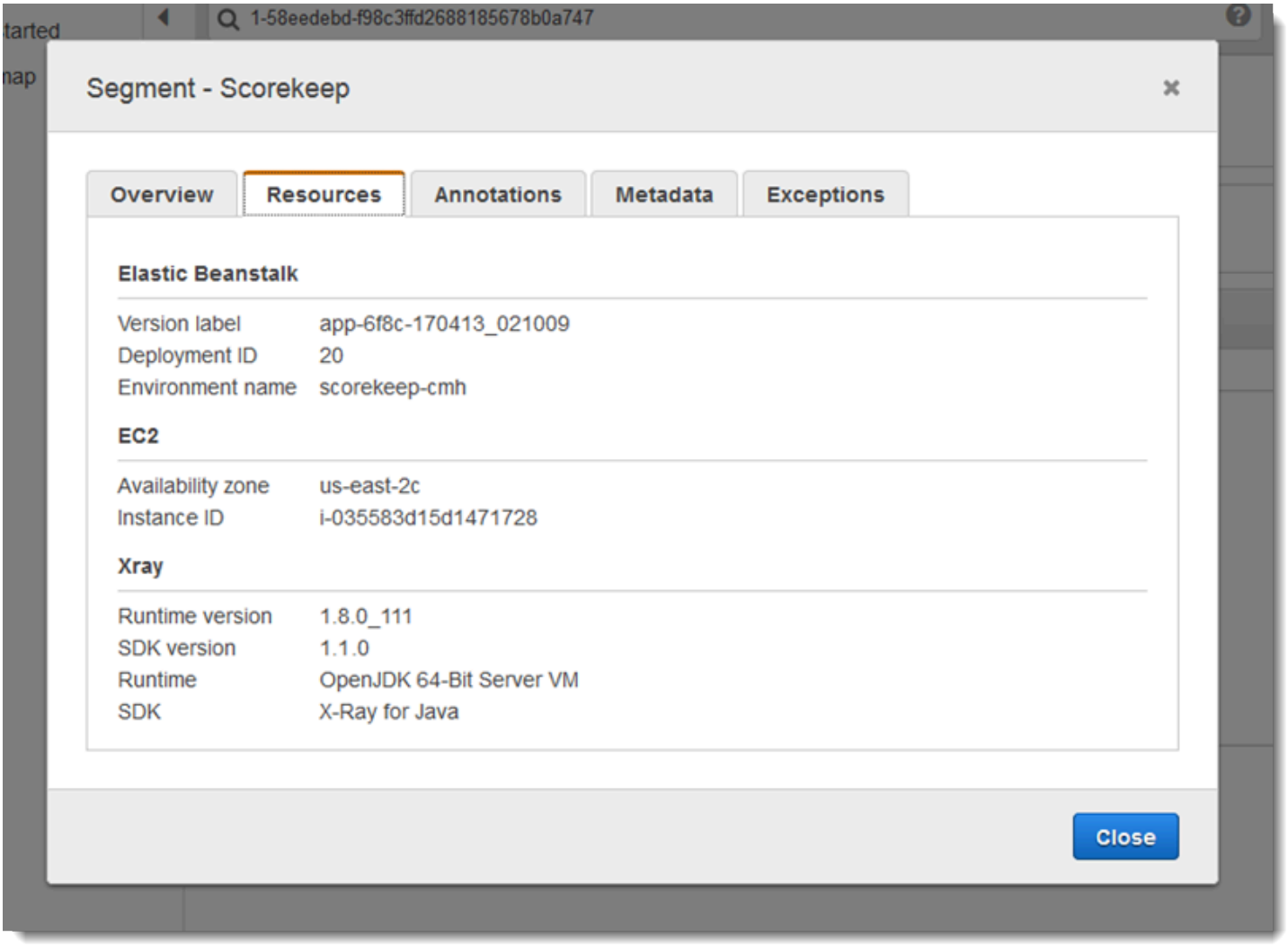
Sections

Service-Plugins

Wird verwendet`plugins`, um Informationen über den Dienst aufzuzeichnen, der Ihre Anwendung hostet.

Plugins

- Amazon EC2 — `EC2Plugin` fügt die Instance-ID, die Availability Zone und die CloudWatch Logs-Gruppe hinzu.
- Elastic Beanstalk — `ElasticBeanstalkPlugin` fügt den Umgebungsnamen, die Versionsbezeichnung und die Bereitstellungs-ID hinzu.
- Amazon ECS — `ECSPlugin` fügt die Container-ID hinzu.
- Amazon EKS — `EKSPlugin` fügt die Container-ID, den Clusternamen, die Pod-ID und die CloudWatch Logs-Gruppe hinzu.



Um ein Plugin zu verwenden, rufen Sie `withPlugin` im `AWSXRayRecorderBuilder` auf.

Example src/main/java/scorekeep/WebConfig.java — Rekorder

```
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());
  }
}
```

Das SDK verwendet auch Plugin-Einstellungen, um das `origin` Feld im Segment festzulegen. Dies gibt den AWS Ressourcentyp an, auf dem Ihre Anwendung ausgeführt wird. Wenn Sie mehrere Plugins verwenden, verwendet das SDK die folgende Auflösungsreihenfolge, um den Ursprung zu bestimmen: ElasticBeanstalk > EKS > ECS > EC2.

Regeln für die Probenahme

Das SDK verwendet die Sampling-Regeln, die Sie in der X-Ray-Konsole definieren, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen. Die Standardregel verfolgt die erste Anfrage jede Sekunde und fünf Prozent aller weiteren Anfragen aller Dienste, die Traces an X-Ray senden. Erstellen Sie zusätzliche Regeln in der X-Ray-Konsole, um die Menge der aufgezeichneten Daten für jede Ihrer Anwendungen anzupassen.

Das SDK wendet benutzerdefinierte Regeln in der Reihenfolge an, in der sie definiert sind. Wenn eine Anfrage mehreren benutzerdefinierten Regeln entspricht, wendet das SDK nur die erste Regel an.

> **Note**
>
> Wenn das SDK X-Ray nicht erreichen kann, um Sampling-Regeln abzurufen, kehrt es zu einer lokalen Standardregel zurück, die die erste Anfrage pro Sekunde und fünf Prozent aller zusätzlichen Anfragen pro Host vorsieht. Dies kann passieren, wenn der Host nicht berechtigt ist APIs, Sampling aufzurufen, oder wenn er keine Verbindung zum X-Ray-Daemon herstellen kann, der als TCP-Proxy für API-Aufrufe durch das SDK fungiert.

Sie können das SDK auch so konfigurieren, dass Sampling-Regeln aus einem JSON-Dokument geladen werden. Das SDK kann lokale Regeln als Backup für Fälle verwenden, in denen X-Ray Sampling nicht verfügbar ist, oder ausschließlich lokale Regeln verwenden.

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
  }
}
```

In diesem Beispiel werden eine benutzerdefinierte Regel und eine Standardregel definiert. Die benutzerdefinierte Regel wendet eine Stichprobenrate von fünf Prozent an, ohne dass eine Mindestanzahl von Anfragen für Pfade verfolgt werden muss. `/api/move/` Die Standardregel verfolgt die erste Anfrage jede Sekunde und 10 Prozent der weiteren Anfragen.

Der Nachteil der lokalen Definition von Regeln besteht darin, dass das feste Ziel von jeder Instanz des Rekorders unabhängig angewendet wird, anstatt vom X-Ray-Dienst verwaltet zu werden. Wenn Sie mehr Hosts bereitstellen, wird die feste Rate vervielfacht, wodurch es schwieriger wird, die Menge der aufgezeichneten Daten zu kontrollieren.

Wenn aktiviert AWS Lambda, können Sie die Samplerate nicht ändern. Wenn Ihre Funktion von einem instrumentierten Dienst aufgerufen wird, werden Aufrufe, die Anfragen generierten, die von diesem Dienst abgetastet wurden, von Lambda aufgezeichnet. Wenn aktives Tracing aktiviert ist und kein Tracing-Header vorhanden ist, trifft Lambda die Stichprobenentscheidung.

Um Sicherungsregeln in Spring bereitzustellen, konfigurieren Sie den globalen Recorder mit einer `CentralizedSamplingStrategy` in einer Konfigurationsklasse.

Example src/main/java/myapp/WebConfig.java — Rekorder-Konfiguration

```
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());
}
```

Für Tomcat fügen Sie einen Listener hinzu, der `ServletContextListener` erweitert, und registrieren den Listener in der Bereitstellungsbeschreibung.

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>
```

Um nur lokale Regeln zu verwenden, ersetzen Sie die `CentralizedSamplingStrategy` durch eine `LocalizedSamplingStrategy`.

```
builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));
```

Protokollierung

Standardmäßig gibt das SDK Meldungen `ERROR` auf -Ebene in Ihre Anwendungsprotokolle aus. Sie können die Protokollierung auf Debug-Ebene im SDK aktivieren, um detailliertere Protokolle in Ihrer Anwendungsprotokolldatei auszugeben. Gültige Protokollebenen sind`DEBUG`,, `INFO``WARN`, `ERROR` und. `FATAL` `FATAL`Bei der Protokollebene werden alle Protokollmeldungen stummgeschaltet, da das SDK nicht auf fataler Ebene protokolliert.

Example application.properties

Legen Sie die Protokollierungsebene mit der `logging.level.com.amazonaws.xray`-Eigenschaft fest.

```
logging.level.com.amazonaws.xray = DEBUG
```

Verwenden Sie Debug-Protokolle, um Probleme wie nicht geschlossene Untersegmente zu identifizieren, wenn Sie Untersegmente manuell generieren.

Trace-ID-Injection in Protokolle

Wenn Sie den Protokollanweisungen Ihre aktuelle vollqualifizierte Trace-ID zur Verfügung stellen möchten, können Sie die ID in den zugeordneten Diagnosekontext (MDC) einfügen. Über die `SegmentListener`-Schnittstelle werden Methoden während der Ereignisse des Segmentlebenszyklus aus dem X-Ray-Recorder aufgerufen. Wenn ein Segment oder Teilsegment beginnt, wird die qualifizierte Trace-ID mit dem Schlüssel `AWS-XRAY-TRACE-ID` in den MDC injiziert. Wenn dieses Segment endet, wird der Schlüssel aus dem MDC entfernt. Dadurch wird die Trace-ID

der verwendeten Protokollierungsbibliothek verfügbar gemacht. Wenn ein Teilsegment endet, wird seine übergeordnete ID in den MDC injiziert.

Example vollqualifizierte Trace-ID

Die vollqualifizierte ID wird als `TraceID@EntityID` dargestellt.

```
1-5df42873-011e96598b447dfca814c156@541b3365be3dafc3
```

Diese Funktion funktioniert mit Java-Anwendungen, die mit dem AWS X-Ray SDK for Java instrumentiert sind, und unterstützt die folgenden Logging-Konfigurationen:

- SLF4J Frontend-API mit Logback-Backend
- SLF4J Frontend-API mit Log4J2-Backend
- Log4J2 Frontend-API mit Log4J2-Backend

In den folgenden Registerkarten finden Sie die Anforderungen jedes Frontends und jedes Backends.

SLF4J Frontend

1. Fügen Sie Ihrem Projekt die folgende Maven-Abhängigkeit hinzu.

   ```
   <dependency>
       <groupId>com.amazonaws</groupId>
       <artifactId>aws-xray-recorder-sdk-slf4j</artifactId>
       <version>2.11.0</version>
   </dependency>
   ```

2. Schließen Sie beim Erstellen von `AWSXRayRecorder` die `withSegmentListener`-Methode ein. Dadurch wird eine `SegmentListener` Klasse hinzugefügt, die automatisch einen neuen Trace in den J MDC einfügt. IDs SLF4

   Der `SegmentListener` akzeptiert eine optionale Zeichenfolge als Parameter, um das Präfix der Log-Anweisung zu konfigurieren. Das Präfix kann auf folgende Weise konfiguriert werden:

   - Keine — Verwendet das `AWS-XRAY-TRACE-ID` Standardpräfix.
   - Leer — Verwendet eine leere Zeichenfolge (z. B.`""`).
   - Benutzerdefiniert — Verwendet ein benutzerdefiniertes Präfix, wie es in der Zeichenfolge definiert ist.

Example **`AWSXRayRecorderBuilder`**-Anweisung

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new SLF4JSegmentListener("CUSTOM-
PREFIX"));
```

Log4J2 front end

1. Fügen Sie Ihrem Projekt die folgende Maven-Abhängigkeit hinzu.

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-log4j</artifactId>
    <version>2.11.0</version>
</dependency>
```

2. Schließen Sie beim Erstellen von `AWSXRayRecorder` die `withSegmentListener`-Methode ein. Dadurch wird eine `SegmentListener` Klasse hinzugefügt, die automatisch einen neuen vollqualifizierten Trace IDs in den SLF4 J MDC einfügt.

   Der `SegmentListener` akzeptiert eine optionale Zeichenfolge als Parameter, um das Präfix der Log-Anweisung zu konfigurieren. Das Präfix kann auf folgende Weise konfiguriert werden:

   - Keine — Verwendet das `AWS-XRAY-TRACE-ID` Standardpräfix.
   - Leer — Verwendet eine leere Zeichenfolge (z. B.`""`) und entfernt das Präfix.
   - Benutzerdefiniert — Verwendet das in der Zeichenfolge definierte benutzerdefinierte Präfix.

   Example **`AWSXRayRecorderBuilder`**-Anweisung

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new Log4JSegmentListener("CUSTOM-
PREFIX"));
```

Logback backend

Um die Trace-ID in die Protokollereignisse einzufügen, müssen Sie das `PatternLayout` des Loggers ändern, der jede Protokollierungsanweisung formatiert.

1. Suchen Sie, wo das `patternLayout` konfiguriert ist. Sie können dies programmgesteuert oder über eine XML-Konfigurationsdatei erreichen. Weitere Informationen finden Sie unter Logback-Konfiguration.
2. Fügen Sie `%X{AWS-XRAY-TRACE-ID}` an einer beliebigen Stelle in `patternLayout` ein, um die Trace-ID in zukünftige Protokollierungsanweisungen einzufügen. `%X{}` gibt an, dass Sie mit dem aus dem MDC bereitgestellten Schlüssel einen Wert abrufen. Weitere Informationen zu PatternLayouts In Logback finden Sie unter PatternLayout.

Log4J2 backend

1. Suchen Sie, wo das `patternLayout` konfiguriert ist. Sie können dies programmgesteuert oder über eine im XML-, JSON-, YAML- oder Eigenschaftenformat geschriebene Konfigurationsdatei erreichen.

   Weitere Informationen zum Konfigurieren von Log4J2 über eine Konfigurationsdatei finden Sie unter Konfiguration.

   Weitere Informationen zur programmgesteuerten Konfiguration von Log4J2 finden Sie unter Programmatische Konfiguration.
2. Fügen Sie `%X{AWS-XRAY-TRACE-ID}` an einer beliebigen Stelle in `PatternLayout` ein, um die Trace-ID in zukünftige Protokollierungsanweisungen einzufügen. `%X{}` gibt an, dass Sie mit dem aus dem MDC bereitgestellten Schlüssel einen Wert abrufen. Weitere Informationen PatternLayouts zu Log4J2 finden Sie unter Pattern Layout.

Beispiel für Trace-ID-Injection

Im Folgenden wird eine `PatternLayout`-Zeichenfolge angezeigt, die geändert wurde, sodass die Trace-ID enthalten ist. Die Trace-ID wird nach dem Thread-Namen (`%t`) und vor der Protokollebene (`%-5p`) ausgegeben.

Example **`PatternLayout`** mit ID-Injection

```
%d{HH:mm:ss.SSS} [%t] %X{AWS-XRAY-TRACE-ID} %-5p %m%n
```

AWS X-Ray druckt automatisch den Schlüssel und die Trace-ID in der Protokollanweisung aus, um das Parsen zu vereinfachen. Im Folgenden wird eine Protokollanweisung dargestellt, die das modifizierte `PatternLayout` verwendet.

Example Protokollanweisung mit ID-Injection

```
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
```

Die Protokollierungsnachricht selbst ist im Muster `%m` eingebettet und wird beim Aufruf des Loggers festgelegt.

Segment-Listeners

Segment-Listener sind eine Schnittstelle zum Abfangen von Lebenszyklusereignissen wie dem Anfang und Ende von Segmenten, die von der erzeugt werden. `AWSXRayRecorder` Die Implementierung einer Segment-Listener-Ereignisfunktion könnte darin bestehen, allen Teilsegmenten dieselbe Anmerkung hinzuzufügen, wenn sie mit `onBeginSubsegment` erstellt werden, eine Meldung zu protokollieren, nachdem jedes Segment mit `afterEndSegment` an den Daemon gesendet wurde, oder von SQL Interceptors mit `beforeEndSubsegment` gesendete Abfragen aufzuzeichnen, um zu überprüfen, ob das Teilsegment eine SQL-Abfrage darstellt, wobei zusätzliche Metadaten hinzugefügt werden, falls dies der Fall ist.

Die vollständige Liste der `SegmentListener` Funktionen finden Sie in der Dokumentation für das AWS X-Ray Recorder SDK for Java API.

Das folgende Beispiel zeigt, wie Sie allen Teilsegmenten bei der Erstellung eine konsistente Anmerkung mit `onBeginSubsegment` hinzufügen und mit `afterEndSegment` eine Protokollmeldung am Ende jedes Segments drucken.

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());
    }
}
```

Dieser benutzerdefinierte Segment-Listener wird dann beim Erstellen des `AWSXRayRecorder` referenziert.

Example AWSXRayRecorderBuilder Aussage

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new MySegmentListener());
```

Umgebungsvariablen

Sie können Umgebungsvariablen verwenden, um das X-Ray SDK for Java zu konfigurieren. Das SDK unterstützt die folgenden Variablen.

- `AWS_XRAY_CONTEXT_MISSING`— Stellen Sie diese Option ein`RUNTIME_ERROR`, um Ausnahmen auszulösen, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, obwohl kein Segment geöffnet ist.

  Zulässige Werte

  - `RUNTIME_ERROR`— Löst eine Laufzeitausnahme aus.
  - `LOG_ERROR`— Fehler protokollieren und fortfahren (Standard).
  - `IGNORE_ERROR`— Fehler ignorieren und fortfahren.

  Fehler im Zusammenhang mit fehlenden Segmenten oder Untersegmenten können auftreten, wenn Sie versuchen, einen instrumentierten Client in Startcode zu verwenden, der ausgeführt wird, wenn keine Anfrage geöffnet ist, oder in Code, der einen neuen Thread erzeugt.
- `AWS_XRAY_DAEMON_ADDRESS`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest. Standardmäßig verwendet `127.0.0.1:2000` das SDK sowohl Trace-Daten (UDP) als auch

Sampling-Daten (TCP). Verwenden Sie diese Variable, wenn Sie den Daemon so konfiguriert haben, dass er auf einem anderen Port lauscht oder wenn er auf einem anderen Host läuft.

Format

- Derselbe Port — *address*:*port*
- Verschiedene Anschlüsse — tcp:*address*:*port* udp:*address*:*port*

- AWS_LOG_GROUP— Legen Sie den Namen einer Protokollgruppe auf eine Protokollgruppe fest, die Ihrer Anwendung zugeordnet ist. Wenn Ihre Protokollgruppe dasselbe AWS Konto und dieselbe Region wie Ihre Anwendung verwendet, sucht X-Ray anhand dieser angegebenen Protokollgruppe automatisch nach den Segmentdaten Ihrer Anwendung. Weitere Informationen zu Protokollgruppen finden Sie unter Arbeiten mit Protokollgruppen und Streams.
- AWS_XRAY_TRACING_NAME— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet. Überschreibt den für die Segmentbenennungsstrategie des Servlet-Filters festgelegten Dienstnamen.

Umgebungsvariablen überschreiben äquivalente Systemeigenschaften und Werte in Code.

Systemeigenschaften

Sie können Systemeigenschaften als JVM-spezifische Alternative zu Umgebungsvariablen verwenden. Das SDK unterstützt die folgenden Eigenschaften:

- com.amazonaws.xray.strategy.tracingName— EntsprichtAWS_XRAY_TRACING_NAME.
- com.amazonaws.xray.emitters.daemonAddress— EntsprichtAWS_XRAY_DAEMON_ADDRESS.
- com.amazonaws.xray.strategy.contextMissingStrategy— EntsprichtAWS_XRAY_CONTEXT_MISSING.

Wenn eine Systemeigenschaft und die entsprechende Umgebungsvariable eingerichtet sind, wird der Wert der Umgebungsvariablen verwendet. Bei beiden Methoden werden Werte in Code überschrieben.

Nachverfolgung eingehender Anfragen mit dem X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können das X-Ray SDK verwenden, um eingehende HTTP-Anfragen zu verfolgen, die Ihre Anwendung auf einer EC2 Instance in Amazon EC2 oder Amazon ECS bearbeitet. AWS Elastic Beanstalk

Verwenden Sie einen `Filter`, um eingehende HTTP-Anforderungen zu instrumentieren. Wenn Sie den X-Ray-Servlet-Filter zu Ihrer Anwendung hinzufügen, erstellt das X-Ray SDK for Java ein Segment für jede gesampelte Anfrage. Dieses Segment umfasst Dauer, Methode und Status der HTTP-Anforderung. Die zusätzliche Instrumentierung schafft Untersegmente zu diesem Segment.

> **Note**
>
> Für AWS Lambda Funktionen erstellt Lambda für jede abgetastete Anfrage ein Segment. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Jedes Segment hat einen Namen, der Ihre Anwendung in der Service Map identifiziert. Das Segment kann statisch benannt werden, oder Sie können das SDK so konfigurieren, dass es dynamisch auf der Grundlage des Host-Headers in der eingehenden Anfrage benannt wird. Mit der dynamischen Benennung können Sie Traces auf der Grundlage des Domainnamens in der Anfrage gruppieren und einen Standardnamen anwenden, wenn der Name nicht einem erwarteten Muster entspricht (z. B. wenn der Host-Header gefälscht ist).

> **Weitergeleitete Anfragen**
>
> Wenn ein Load Balancer oder ein anderer Vermittler eine Anfrage an Ihre Anwendung weiterleitet, nimmt X-Ray die Client-IP aus dem `X-Forwarded-For` Header in der Anfrage und nicht aus der Quell-IP im IP-Paket. Die Client-IP, die für eine weitergeleitete Anfrage aufgezeichnet wird, kann gefälscht sein und sollte daher nicht als vertrauenswürdig eingestuft werden.

Wenn eine Anfrage weitergeleitet wird, legt das SDK ein zusätzliches Feld im Segment fest, um dies anzuzeigen. Wenn das Segment das Feld enthält, das auf `x_forwarded_for` gesetzt ist`true`, wurde die Client-IP aus dem `X-Forwarded-For` Header in der HTTP-Anfrage übernommen.

Der Meldungshandler erzeugt für jede eingehende Anforderung ein Segment mit einem `http`-Block, der die folgenden Informationen enthält:

- HTTP-Methode — GET, POST, PUT, DELETE usw.
- Client-Adresse — Die IP-Adresse des Clients, der die Anfrage gesendet hat.
- Antwortcode — Der HTTP-Antwortcode für die abgeschlossene Anfrage.
- Timing — Die Startzeit (als die Anfrage empfangen wurde) und die Endzeit (als die Antwort gesendet wurde).
- Benutzeragent — Der `user-agent` aus der Anfrage.
- Länge des Inhalts — Die Länge `content-length` der Antwort.

Sections

Hinzufügen eines Ablaufverfolgungsfilters zu Ihrer Anwendung (Tomcat)

Im Fall von Tomcat fügen Sie einen `<filter>` zur `web.xml`-Datei Ihres Projekts hinzu. Legen Sie mit dem `fixedName` Parameter einen Servicenamen fest, damit die erstellten Segmente auf eingehende Anforderungen angewendet werden.

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>
```

Hinzufügen eines Ablaufverfolgungsfilters zu Ihrer Anwendung (Spring)

Für Spring fügen Sie einen `Filter` zu Ihrer `WebConfig`-Klasse hinzu. Übermitteln Sie den Segmentnamen als Zeichenfolge an den `AWSXRayServletFilter`-Konstruktor.

Example src/main/java/myapp/WebConfig.java - Frühling

```
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");
  }
}
```

Konfiguration einer Segmentbenennungsstrategie

AWS X-Ray verwendet einen Dienstnamen, um Ihre Anwendung zu identifizieren und sie von den anderen Anwendungen, Datenbanken, externen AWS Ressourcen und Ressourcen zu unterscheiden

APIs, die Ihre Anwendung verwendet. Wenn das X-Ray SDK Segmente für eingehende Anfragen generiert, zeichnet es den Dienstnamen Ihrer Anwendung im Namensfeld des Segments auf.

Das X-Ray SDK kann Segmente nach dem Hostnamen im HTTP-Anforderungsheader benennen. Dieser Header kann jedoch gefälscht sein, was zu unerwarteten Knoten in Ihrer Service Map führen kann. Um zu verhindern, dass das SDK Segmente aufgrund von Anfragen mit gefälschten Host-Headern falsch benennt, müssen Sie einen Standardnamen für eingehende Anfragen angeben.

Wenn Ihre Anwendung Anfragen für mehrere Domänen bearbeitet, können Sie das SDK so konfigurieren, dass es eine dynamische Benennungsstrategie verwendet, um dies in Segmentnamen widerzuspiegeln. Eine dynamische Benennungsstrategie ermöglicht es dem SDK, den Hostnamen für Anfragen zu verwenden, die einem erwarteten Muster entsprechen, und den Standardnamen auf Anfragen anzuwenden, bei denen dies nicht der Fall ist.

Beispielsweise können Sie über eine einzige Anwendung verfügen, die Anfragen an drei Subdomänen — `www.example.com`api.example.com, und — bedient. `static.example.com` Sie können eine dynamische Benennungsstrategie mit dem Muster verwenden`*.example.com`, um Segmente für jede Subdomain mit einem anderen Namen zu identifizieren, was zu drei Dienstknoten auf der Service-Map führt. Wenn Ihre Anwendung Anfragen mit einem Hostnamen empfängt, der nicht dem Muster entspricht, wird auf der Service Map ein vierter Knoten mit einem von Ihnen angegebenen Fallback-Namen angezeigt.

Wenn Sie denselben Namen für alle Segmente verwenden möchten, geben Sie bei der Initialisierung des Servlet-Filters den Namen Ihrer Anwendung, wie im vorherigen Abschnitt gezeigt, ein. Dies hat den gleichen Effekt wie das Erstellen eines Fixed SegmentNamingStrategydurch Aufrufen `SegmentNamingStrategy.fixed()` und Übergeben an den `AWSXRayServletFilter`Konstruktor.

> **Note**
>
> Sie können den mit der `AWS_XRAY_TRACING_NAME`-Umgebungsvariablen in Code definierten standardmäßigen Dienstnamen überschreiben.

Eine dynamische Benennungsstrategie definiert ein Muster, dem Hostnamen entsprechen sollten, sowie einen Standardnamen, der verwendet wird, wenn der Hostname in der HTTP-Anforderung nicht mit diesem Muster übereinstimmt. Alternativ können Sie `dynamicNamingRecognizedHosts` und `dynamicNamingFallbackName` nutzen, um das Muster und den Standardnamen zu bestimmen und Segmente in Tomcat dynamisch zu benennen.

Example WEB-INF/web.xml – Servlet-Filter mit dynamischer Benennung

```
<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>
```

Erstellen Sie für Spring eine Dynamik, SegmentNamingStrategyindem Sie sie aufrufen`SegmentNamingStrategy.dynamic()`, und übergeben Sie sie an den `AWSXRayServletFilter` Konstruktor.

Example src/main/java/myapp/WebConfig.java — Servlet-Filter mit dynamischer Benennung

```
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"));
  }
}
```

Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Aufrufe tätigt, um Daten AWS-Services zu speichern, in eine Warteschlange zu schreiben oder Benachrichtigungen zu senden, verfolgt das X-Ray SDK for Java die Aufrufe im Downstream in Untersegmenten. Verfolgte Ressourcen AWS-Services und Ressourcen, auf die Sie innerhalb dieser Services zugreifen (z. B. ein Amazon S3 S3-Bucket oder eine Amazon SQS SQS-Warteschlange), werden als Downstream-Knoten auf der Trace-Map in der X-Ray-Konsole angezeigt.

Das X-Ray-SDK SDK for Java instrumentiert automatisch alle AWS-SDK-Clients, wenn Sie die `aws-sdk` und die `aws-sdk-instrumentor` Submodule in Ihren Build aufnehmen. Wenn Sie das Instrumentor-Untermodul nicht einbeziehen, können Sie auswählen, welche Clients Sie instrumentieren möchten, und andere ausschließen.

Um einzelne Clients zu instrumentieren, entfernen Sie das `aws-sdk-instrumentor` Submodul aus Ihrem Build und fügen Sie mithilfe des Client-Builders des Services ein `XRayClient` AS `TracingHandler` auf Ihrem AWS SDK-Client hinzu.

Wenn Sie beispielsweise einen `AmazonDynamoDB`-Client instrumentieren möchten, übergeben Sie einen Ablaufverfolgungshandler an `AmazonDynamoDBClientBuilder`.

Example MyModel.java — DynamoDB-Client

```
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();
...
```

Für alle Dienste können Sie den Namen der aufgerufenen API in der X-Ray-Konsole sehen. Für eine Untergruppe von Diensten fügt das X-Ray SDK dem Segment Informationen hinzu, um die Service Map detaillierter zu gestalten.

Wenn Sie beispielsweise einen Aufruf mit einem instrumentierten DynamoDB-Client tätigen, fügt das SDK den Tabellennamen dem Segment für Aufrufe hinzu, die auf eine Tabelle abzielen. In der Konsole wird jede Tabelle als separater Knoten in der Service Map angezeigt, mit einem generischen DynamoDB-Knoten für Aufrufe, die nicht auf eine Tabelle abzielen.

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements

```
{
  "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",
  }
}
```

Wenn Sie auf benannte Ressourcen zugreifen, werden durch Aufrufe der folgenden Services weitere Knoten in der Service-Übersicht erstellt. Durch Aufrufe, die keinen bestimmten Ressourcen gelten, wird ein generischer Knoten für den Service erstellt.

- Amazon DynamoDB — Tabellenname
- Amazon Simple Storage Service — Bucket und Schlüsselname
- Amazon Simple Queue Service — Name der Warteschlange

Um Downstream-Aufrufe AWS-Services mit AWS SDK für Java 2.2 und höher zu instrumentieren, können Sie das `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` Modul aus Ihrer Build-Konfiguration weglassen. Fügen Sie stattdessen das `aws-xray-recorder-sdk-aws-sdk-v2 module` ein und instrumentieren Sie dann einzelne Clients, indem Sie sie mit einem `TracingInterceptor` konfigurieren.

Example AWS SDK für Java 2.2 und höher — Tracing Interceptor

```
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();
//...
```

Rückverfolgung von Aufrufen an Downstream-HTTP-Webservices mit dem X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Microservices oder öffentliches HTTP aufruft APIs, können Sie die Version von X-Ray SDK für Java verwenden, `HttpClient` um diese Aufrufe zu instrumentieren und die API als Downstream-Service zum Service Graph hinzuzufügen.

Das X-Ray-SDK SDK for Java umfasst `DefaultHttpClient` `HttpClientBuilder` Klassen, die anstelle der HttpComponents Apache-Äquivalente verwendet werden können, um ausgehende HTTP-Aufrufe zu instrumentieren.

- `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`

Diese Bibliotheken befinden sich im `aws-xray-recorder-sdk-apache-http`-Untermodul.

Sie können Ihre vorhandenen Importanweisungen durch das X-Ray-Äquivalent ersetzen, um alle Clients zu instrumentieren, oder den vollqualifizierten Namen verwenden, wenn Sie einen Client für die Instrumentierung bestimmter Clients initialisieren.

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();
    }
  }
```

Wenn Sie einen Aufruf einer Downstream-Web-API instrumentieren, zeichnet das X-Ray SDK for Java ein Untersegment mit Informationen über die HTTP-Anfrage und -Antwort auf. X-Ray verwendet das Untersegment, um ein abgeleitetes Segment für die Remote-API zu generieren.

Example Untersegment für einen nachgelagerten HTTP-Aufruf

```
{
  "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 Abgeleitetes Segment für einen nachgelagerten HTTP-Anruf

```
{
  "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
}
```

Verfolgen von SQL-Abfragen mit dem X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

SQL-Interzeptoren

Instrumentieren Sie SQL-Datenbankabfragen, indem Sie den X-Ray SDK for Java JDBC Interceptor zu Ihrer Datenquellenkonfiguration hinzufügen.

- PostgreSQL – `com.amazonaws.xray.sql.postgres.TracingInterceptor`
- MySQL – `com.amazonaws.xray.sql.mysql.TracingInterceptor`

Diese Interceptors befinden sich im `aws-xray-recorder-sql-postgres-` bzw. `aws-xray-recorder-sql-mysql-Untermodul`. Sie implementieren `org.apache.tomcat.jdbc.pool.JdbcInterceptor` und sind mit Tomcat-Verbindungspools kompatibel.

> **Note**
>
> Aus Sicherheitsgründen zeichnen die SQL Interceptors die SQL-Abfrage selbst innerhalb von Teilsegmenten nicht auf.

Für Spring fügen Sie der Eigenschaftendatei einen Interceptor hinzu und erstellen Sie die Datenquelle mit dem `DataSourceBuilder` von Spring Boot.

Example **`src/main/java/resources/application.properties`** – PostgreSQL-JDBC-Interceptor

```
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`** – Datenquelle

```
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();
  }
...
}
```

Rufen Sie für Tomcat die JDBC-Datenquelle mit einem Verweis `setJdbcInterceptors` auf die Klasse X-Ray SDK for Java auf.

Example **`src/main/myapp/model.java`** – Datenquelle

```
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;");
```

Die Tomcat JDBC Data Source-Bibliothek ist im X-Ray SDK for Java enthalten, aber Sie können sie als bereitgestellte Abhängigkeit von dem Dokument deklarieren, in dem Sie sie verwenden.

Example **`pom.xml`** – JDBC-Datenquelle

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jdbc</artifactId>
  <version>8.0.36</version>
  <scope>provided</scope>
</dependency>
```

Nativer SQL Tracing Decorator

- Fügen Sie Ihre aws-xray-recorder-sdk-sqlAbhängigkeiten hinzu.
- Dekorieren Sie Ihre Datenbank-Datenquelle, Verbindung oder Anweisung.

```
dataSource = TracingDataSource.decorate(dataSource)
connection = TracingConnection.decorate(connection)
statement = TracingStatement.decorateStatement(statement)
preparedStatement = TracingStatement.decoratePreparedStatement(preparedStatement,
 sql)
callableStatement = TracingStatement.decorateCallableStatement(callableStatement,
 sql)
```

Generieren von benutzerdefinierten Untersegmenten mit dem X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Untersegmente erweitern das Segment eines Traces um Details über die Arbeit, die zur Bearbeitung einer Anfrage geleistet wurde. Jedes Mal, wenn Sie einen Anruf mit einem instrumentierten Client tätigen, zeichnet das X-Ray-SDK die in einem Untersegment generierten Informationen auf. Sie können zusätzliche Untersegmente erstellen, um andere Untersegmente zu gruppieren, die Leistung eines Codeabschnitts zu messen oder Anmerkungen und Metadaten aufzuzeichnen.

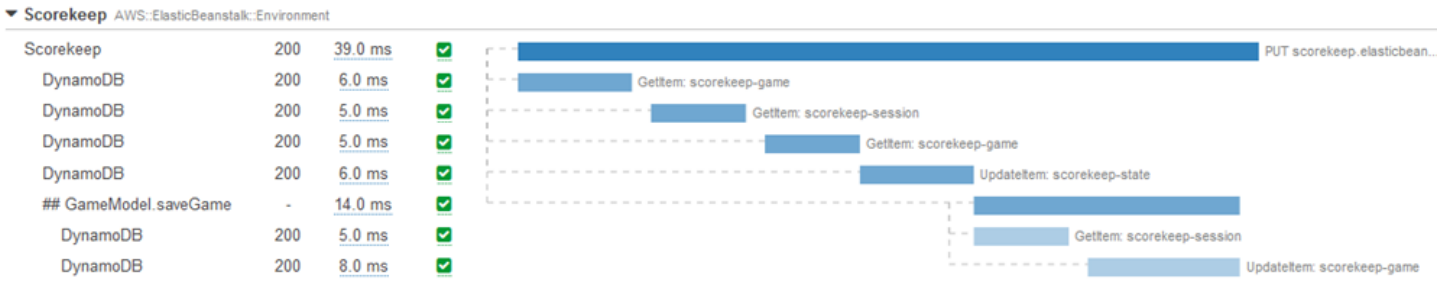
Um Untersegmente zu verwalten, verwenden Sie die Methoden `beginSubsegment` und `endSubsegment`.

Example GameModel.java — benutzerdefiniertes Untersegment

```
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();
    }
  }
```

In diesem Beispiel lädt der Code innerhalb des Untersegments die Spielsitzung aus DynamoDB mit einer Methode auf dem Sitzungsmodell und verwendet den DynamoDB-Mapper AWS SDK für Java des Spiels, um das Spiel zu speichern. Wenn Sie diesen Code in ein Untersegment einbinden, werden die Aufrufe zu DynamoDB-Unterelementen des `Save Game` Untersegments in der Trace-Ansicht in der Konsole.



Wenn der Code in Ihrem Untersegment geprüfte Ausnahmen auslöst, verpacken Sie ihn in einen `try`-Block und rufen Sie `AWSXRay.endSubsegment()` in einem `finally`-Block auf, um sicherzustellen, dass das Untersegment immer geschlossen ist. Wenn ein Untersegment nicht geschlossen ist, kann das übergeordnete Segment nicht abgeschlossen werden und wird nicht an X-Ray gesendet.

Für Code, der keine geprüften Ausnahmen auslöst, können Sie den Code `AWSXRay.CreateSubsegment` als Lambda-Funktion übergeben.

Example Untersegment-Lambda-Funktion

```
import com.amazonaws.xray.AWSXRay;

AWSXRay.createSubsegment("getMovies", (subsegment) -> {
    // function code
});
```

Wenn Sie ein Untersegment innerhalb eines Segments oder eines anderen Untersegments erstellen, generiert das X-Ray SDK for Java eine ID dafür und zeichnet die Start- und Endzeit auf.

Example Untersegment mit Metadaten

```
"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"
    }
  },
```

Bei asynchroner Programmierung und Multithread-Programmierung müssen Sie das Teilsegment manuell an die `endSubsegment()` Methode übergeben, um sicherzustellen, dass es korrekt geschlossen wird, da der X-Ray-Kontext während der asynchronen Ausführung geändert werden kann. Wenn ein asynchrones Teilsegment geschlossen wird, nachdem das übergeordnete Segment geschlossen wurde, streamt diese Methode automatisch das gesamte Segment an den X-Ray-Daemon.

Example Asynchrones Teilsegment

```
@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();
}
```

Hinzufügen von Anmerkungen und Metadaten zu Segmenten mit dem X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können zusätzliche Informationen über Anfragen, die Umgebung oder Ihre Anwendung mit Anmerkungen und Metadaten aufzeichnen. Sie können Anmerkungen und Metadaten zu den Segmenten hinzufügen, die das X-Ray SDK erstellt, oder zu benutzerdefinierten Untersegmenten, die Sie erstellen.

Anmerkungen sind Schlüssel-Wert-Paare mit Zeichenfolgen-, Zahlen- oder booleschen Werten. Anmerkungen sind für die Verwendung mit Filterausdrücken indexiert. Berücksichtigen Sie Anmerkungen, um Daten zur Gruppierung von Ablaufverfolgungen in der Konsole zu verwenden, oder wenn Sie die `GetTraceSummaries`-API aufrufen.

Metadaten sind Schlüssel-Wert-Paare, die Werte beliebigen Typs enthalten können, einschließlich Objekte und Listen, aber nicht für die Verwendung mit Filterausdrücken indexiert sind. Verwenden Sie Metadaten, um zusätzliche Daten aufzuzeichnen, die Sie im Trace speichern möchten, aber nicht für die Suche verwenden müssen.

Zusätzlich zu Anmerkungen und Metadaten können Sie auch Benutzer-ID-Zeichenfolgen in Segmenten aufzeichnen. Benutzer IDs werden in einem separaten Feld in Segmenten aufgezeichnet und für die Verwendung bei der Suche indexiert.

Sections

Anmerkungen mit dem X-Ray SDK for Java aufnehmen

Verwenden Sie Anmerkungen, um Informationen zu Segmenten oder Untersegmenten, die zur Suche indiziert werden sollten, aufzuzeichnen.

Anmerkung zu Anforderungen

- Schlüssel — Der Schlüssel für eine X-Ray-Anmerkung kann bis zu 500 alphanumerische Zeichen enthalten. Sie können keine anderen Leerzeichen oder Symbole als einen Punkt oder Punkt (.) verwenden
- Werte — Der Wert für eine X-Ray-Anmerkung kann bis zu 1.000 Unicode-Zeichen enthalten.
- Die Anzahl der Anmerkungen — Sie können bis zu 50 Anmerkungen pro Spur verwenden.

So zeichnen Sie Anmerkungen auf

1. Eine Referenz des aktuellen Segments oder Untersegments finden Sie unter `AWSXRay`.

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

   oder

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Subsegment;
   ...
   Subsegment document = AWSXRay.getCurrentSubsegment();
   ```

2. Rufen Sie `putAnnotation` mit einem Aktivierungsschlüssel und einem booleschen Wert oder einem Zeichenfolgenwert auf.

   ```
   document.putAnnotation("mykey", "my value");
   ```

   Das folgende Beispiel zeigt, wie Sie `putAnnotation` mit einem String-Schlüssel aufrufen, der einen Punkt und einen booleschen Wert, eine Zahl oder einen String-Wert enthält.

   ```
   document.putAnnotation("testkey.test", "my value");
   ```

Das SDK zeichnet Anmerkungen als Schlüssel-Wert-Paare in einem `annotations`-Objekt im Segmentdokument auf. Wenn `putAnnotation` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Nutzen Sie das `annotation[key]`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen durch Anmerkungen mit bestimmten Werten zu finden.

Example **src/main/java/scorekeep/GameModel.java**— Anmerkungen und Metadaten

```
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();
    }
  }
```

Metadaten mit dem X-Ray SDK for Java aufzeichnen

Verwenden Sie Metadaten, um Segment- oder Untersegmentinformationen aufzuzeichnen, die nicht zur Suche indiziert werden müssen. Metadatenwerte sind Zeichenfolgen, Zahlen, boolesche Werte oder andere Objekte, die in Form eines JSON-Objekts oder eines Arrays angeordnet sein können.

So zeichnen Sie Metadaten auf

1. Eine Referenz des aktuellen Segments oder Untersegments finden Sie unter `AWSXRay`.

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

   oder

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Subsegment;
   ...
   Subsegment document = AWSXRay.getCurrentSubsegment();
   ```

2. Rufen Sie `putMetadata` mit einem String-Namespace, einem Aktivierungsschlüssel sowie einem booleschen Wert, einer Zahl, einer Zeichenfolge oder einem Objektwert auf.

   ```
   document.putMetadata("my namespace", "my key", "my value");
   ```

   oder

   Rufen Sie `putMetadata` nur mit einem Aktivierungsschlüssel und einem Wert auf.

   ```
   document.putMetadata("my key", "my value");
   ```

Wenn Sie keinen Namespace angeben, verwendet SDK `default`. Wenn `putMetadata` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Example **`src/main/java/scorekeep/GameModel.java`**— Anmerkungen und Metadaten

```
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();
    }
  }
```

Benutzer IDs mit dem X-Ray SDK for Java aufzeichnen

Zeichnen Sie Segmente des Benutzers IDs auf Anfrage auf, um den Benutzer zu identifizieren, der die Anfrage gesendet hat.

Um den Benutzer aufzuzeichnen IDs

1. Eine Referenz des aktuellen Segments finden Sie unter `AWSXRay`.

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

2. Rufen Sie `setUser` mit einer Zeichenfolgen-ID des Benutzers auf, der die Anforderung gesendet hat.

   ```
   document.setUser("U12345");
   ```

Sie können `setUser` in Ihrem Controller aufrufen, um die Benutzer-ID aufzuzeichnen, sobald die Anwendung mit der Bearbeitung einer Anfrage beginnt. Wenn Sie das Segment nur zur Einrichtung der Benutzer-ID verwenden, können Sie die Aufrufe in einer einzelnen Zeile anordnen.

Example src/main/java/scorekeep/MoveController.java — Benutzer-ID

```
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);
  }
```

Nutzen Sie das `user`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen einer Benutzer-ID zu finden.

AWS X-Ray Metriken für das X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

In diesem Thema werden der AWS X-Ray Namespace, die Metriken und die Dimensionen beschrieben. Sie können das X-Ray SDK for Java verwenden, um CloudWatch Amazon-Metriken ohne Stichproben aus Ihren gesammelten X-Ray-Segmenten zu veröffentlichen. Diese Metriken werden von der Start- und Endzeit des Segments sowie den Status-Flags für Fehler, Ausfall und Ablehnung abgeleitet. Mit diesen Trace-Metriken können Sie Wiederholungen und Abhängigkeitsprobleme in Teilsegmenten anzeigen.

CloudWatch ist ein Metrik-Repository. Eine Metrik ist das grundlegende Konzept von Datenpunkten CloudWatch und stellt eine zeitlich geordnete Menge von Datenpunkten dar. Sie (oder AWS-

Services) veröffentlichen Metrikdatenpunkte in CloudWatch und rufen Statistiken über diese Datenpunkte als geordneten Satz von Zeitreihendaten ab.

Metriken werden eindeutig durch einen Namen, ein Namespace und eine oder mehrere Dimensionen definiert. Jeder Datenpunkt verfügt über einen Zeitstempel und optional über eine Maßeinheit. Wenn Sie Statistiken anfordern, wird der zurückgegebene Datenstrom durch den Namespace, den Metrik-Namen und die Dimension identifiziert.

Weitere Informationen zu CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch Röntgenmetriken

Der `ServiceMetrics/SDK`-Namespace enthält die folgenden Metriken.

Metrik	Verfügbare Statistiken	Description	Einheiten
`Latency`	Durchschnitt, Minimum, Maximum, Anzahl	Die Differenz zwischen der Start- und Endzeit Durchschnitt, Minimum und Maximum beschreiben Betriebslatenz. „Anzahl“ beschreibt die Anzahl der Aufrufe.	Millisekunden
`ErrorRate`	Durchschnitt, Summe	Die Rate der Anforderungen, die mit dem Statuscode „`4xx Client Error`“ fehlgeschlagen sind, was zu einem Fehler führt.	Prozent
`FaultRate`	Durchschnitt, Summe	Die Rate der Traces, die mit dem Statuscode „`5xx Server Error`“ fehlgesch	Prozent

Metrik	Verfügbare Statistiken	Description	Einheiten
		lagen sind, was zu einem Fehler führte.	
`ThrottleRate`	Durchschnitt, Summe	Die Rate der abgelehnten Traces, die einen 429-Statuscode zurückgeben. Dies ist eine Teilmenge der `ErrorRate` -Metrik.	Prozent
`OkRate`	Durchschnitt, Summe	Die Rate der verfolgten Anforderungen, die zu einem OK-Statuscode führen.	Prozent

CloudWatch Röntgenabmessungen

Verwenden Sie die Dimensionen in der folgenden Tabelle, um die für Ihre Java Anwendungen mit Röntgeninstrumenten zurückgegebenen Messwerte zu verfeinern.

Dimension	Description
`ServiceType`	Der Service-Typ, z. B. `AWS::EC2::Instance` oder `NONE`, falls nicht bekannt.
`ServiceName`	Der kanonische Name für den Service.

CloudWatch X-Ray-Metriken aktivieren

Gehen Sie wie folgt vor, um Trace-Metriken in Ihrer instrumentierten Java Anwendung zu aktivieren.

So konfigurieren Sie Trace-Metriken

1. Fügen Sie das `aws-xray-recorder-sdk-metrics` Paket als Apache Maven Abhängigkeit hinzu. Weitere Informationen finden Sie unter X-Ray SDK for Java Java-Submodule.

2. Aktivieren Sie ein neues `MetricsSegmentListener()` als Teil des globalen Recorder-Builds.

   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. Stellen Sie den CloudWatch Agenten bereit, um Metriken mithilfe von Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS) oder Amazon Elastic Kubernetes Service (Amazon EKS) zu sammeln:

   - Informationen zur Konfiguration von Amazon EC2 finden Sie unter Installation des CloudWatch Agenten.
   - Informationen zur Konfiguration von Amazon ECS finden Sie unter Überwachen von Amazon ECS-Containern mithilfe von Container Insights.
   - Informationen zur Konfiguration von Amazon EKS finden Sie unter Installieren des CloudWatch Agenten mithilfe des Amazon CloudWatch Observability EKS-Add-ons.

4. Konfigurieren Sie das SDK für die Kommunikation mit dem CloudWatch Agenten. Standardmäßig kommuniziert das SDK mit dem CloudWatch Agenten über die

Adresse`127.0.0.1`. Sie können alternative Adressen konfigurieren, indem Sie die Umgebungsvariable oder die Java-Eigenschaft auf `address:port` festlegen.

Example Umgebungsvariable

```
AWS_XRAY_METRICS_DAEMON_ADDRESS=address:port
```

Example Java-Eigenschaft

```
com.amazonaws.xray.metrics.daemonAddress=address:port
```

So überprüfen Sie die Konfiguration

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Öffnen Sie die Registerkarte Metriken, um den Zustrom Ihrer Metriken zu überwachen.
3. (Optional) Öffnen Sie in der CloudWatch Konsole auf der Registerkarte Protokolle die `ServiceMetricsSDK` Protokollgruppe. Suchen Sie nach einem Protokolldatenstrom, der den Host-Metriken entspricht, und bestätigen Sie die Protokollmeldungen.

Übermitteln von Segmentkontext zwischen Threads in einer Multithread-Anwendung

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Beim Erstellen eines neuen Threads in Ihrer Anwendung behält `AWSXRayRecorder` eine Referenz zur aktuellen Segment- oder Untersegment-Entity nicht bei. Wenn Sie im neuen Thread einen

instrumentierten Client verwenden, versucht das SDK, in ein Segment zu schreiben, das nicht existiert, was zu einem führt. SegmentNotFoundException

Um zu vermeiden, dass während der Entwicklung Ausnahmen ausgelöst werden, können Sie den Rekorder so konfigurieren ContextMissingStrategy, dass er stattdessen einen Fehler protokollieren soll. Sie können die Strategie im Code mit einer SetContextMissingStrategyUmgebungsvariablen oder einer Systemeigenschaft konfigurieren oder entsprechende Optionen konfigurieren.

Eine Möglichkeit zur Behebung des Fehlers ist die Verwendung eines neuen Segments, indem Sie beginSegment aufrufen, wenn Sie den Thread starten, und endSegment, wenn Sie ihn schließen. Dies funktioniert, wenn Sie Code instrumentieren, der nicht als Reaktion auf eine HTTP-Anforderung ausgeführt wird, z. B. Code, der beim Starten Ihrer Anwendung ausgeführt wird.

Wenn Sie mehrere Threads zur Verarbeitung eingehender Anfragen verwenden, können Sie das aktuelle Segment oder Untersegment an den neuen Thread übergeben und für die globale Aufzeichnung bereitstellen. Auf diese Weise wird sichergestellt, dass die im neuen Thread aufgezeichneten Informationen mit demselben Segment verknüpft werden wie die übrigen zu dieser Anfrage aufgezeichneten Informationen. Sobald das Segment im neuen Thread verfügbar ist, können Sie jedes Runnable mit Zugriff auf den Kontext dieses Segments mithilfe der `segment.run(() -> { ... })` Methode ausführen.

Ein Beispiel finden Sie unter Verwenden instrumentierter Clients in Auftragnehmer-Threads.

X-Ray mit asynchroner Programmierung verwenden

Das X-Ray SDK for Java kann in asynchronen Java-Programmen mit SegmentContextExecutorsverwendet werden. Das SegmentContextExecutor implementiert das Executor-Interface, was bedeutet, dass es an alle asynchronen Operationen von a übergeben werden kann. CompletableFuture Dadurch wird sichergestellt, dass alle asynchronen Operationen mit dem richtigen Segment in ihrem Kontext ausgeführt werden.

Example Beispiel App.java: Übergabe an 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 mit Spring und dem X-Ray SDK for Java

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

In diesem Thema wird beschrieben, wie Sie das X-Ray SDK und das Spring Framework verwenden, um Ihre Anwendung zu instrumentieren, ohne ihre Kernlogik zu ändern. Das bedeutet, dass es jetzt eine nichtinvasive Methode zur Instrumentierung Ihrer Anwendungen gibt, die remote ausgeführt werden. AWS

So aktivieren Sie AOP in Spring

1. Konfigurieren von Spring
2. Fügen Sie Ihrer Anwendung einen Tracing-Filter hinzu
3. Kommentieren Ihres Codes oder implementieren einer Schnittstelle
4. Aktivieren von X-Ray in Ihrer Anwendung

Konfigurieren von Spring

Sie können Maven oder Gradle verwenden, um Spring zu konfigurieren, um AOP für die Instrumentierung Ihrer Anwendung verwenden zu können.

Wenn Sie Ihre Anwendung mit Maven erstellen, fügen Sie die folgende Abhängigkeit in Ihrer `pom.xml`-Datei hinzu.

```
<dependency>
     <groupId>com.amazonaws</groupId>
     <artifactId>aws-xray-recorder-sdk-spring</artifactId>
     <version>2.11.0</version>
</dependency>
```

Für Gradle fügen Sie die folgende Abhängigkeit in Ihre `build.gradle`-Datei ein.

```
compile 'com.amazonaws:aws-xray-recorder-sdk-spring:2.11.0'
```

Spring Boot konfigurieren

Wenn Sie Spring Boot verwenden, fügen Sie zusätzlich zu der im vorherigen Abschnitt beschriebenen Spring-Abhängigkeit die folgende Abhängigkeit hinzu, sofern sie nicht bereits in Ihrem Klassenpfad enthalten ist.

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'
```

Hinzufügen eines Tracing-Filters zu Ihrer Anwendung

Fügen Sie Ihrer `WebConfig` Klasse eine `Filter` hinzu. Übermitteln Sie den Segmentnamen als Zeichenfolge an den `AWSXRayServletFilter`-Konstruktor. Weitere Informationen zur Verfolgung von Filtern und zur Instrumentierung eingehender Anfragen finden Sie unter. Nachverfolgung eingehender Anfragen mit dem X-Ray SDK for Java

Example src/main/java/myapp/WebConfig.java — Frühling

```
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");
  }
}
```

Jakarta-Unterstützung

Spring 6 verwendet Jakarta anstelle von Javax für seine Enterprise Edition. Um diesen neuen Namespace zu unterstützen, hat X-Ray einen parallel Satz von Klassen erstellt, die in ihrem eigenen Jakarta-Namespace leben.

Ersetzen Sie für die Filterklassen durch. `javax` `jakarta` Wenn Sie eine Segmentbenennungsstrategie konfigurieren, fügen Sie `jakarta` vor der Benennungsstrategie den Klassennamen hinzu, wie im folgenden Beispiel:

```
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"));
    }
}
```

Kommentieren Ihres Codes oder Implementieren einer Schnittstelle

Ihre Klassen müssen entweder mit der `@XRayEnabled` Anmerkung versehen sein oder die `XRayTraced` Schnittstelle implementieren. Damit wird das AOP-System angewiesen, die Funktionen der betroffenen Klasse für die X-Ray-Instrumentierung zu kapseln.

X-Ray in Ihrer Anwendung aktivieren

Um X-Ray Tracing in Ihrer Anwendung zu aktivieren, muss Ihr Code die abstrakte Klasse erweitern, `BaseAbstractXRayInterceptor` indem er die folgenden Methoden überschreibt.

- `generateMetadata`— Diese Funktion ermöglicht die Anpassung der Metadaten, die an den Trace der aktuellen Funktion angehängt sind. Standardmäßig wird der Klassenname der ausgeführten Funktion in den Metadaten aufgezeichnet. Sie können weitere Daten hinzufügen, wenn Sie zusätzliche Informationen benötigen.
- `xrayEnabledClasses`— Diese Funktion ist leer und sollte es auch bleiben. Sie dient als Host für ein Pointcut, das den Interceptor anweist, welche Methoden gekapselt werden sollen. Definieren Sie das Pointcut, indem Sie angeben, welche der Klassen mit `@XRayEnabled` kommentiert sind, um ein Tracing durchzuführen. Die folgende pointcut-Anweisung weist den Interceptor an, alle Controller-Beans einzukapseln, die mit dem Kommentar `@XRayEnabled` gekennzeichnet sind.

  ```
  @Pointcut(“@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)”)
  ```

Wenn Ihr Projekt Spring Data JPA verwendet, sollten Sie eine Erweiterung von `AbstractXRayInterceptor` anstelle von in Betracht ziehen. `BaseAbstractXRayInterceptor`

Beispiel

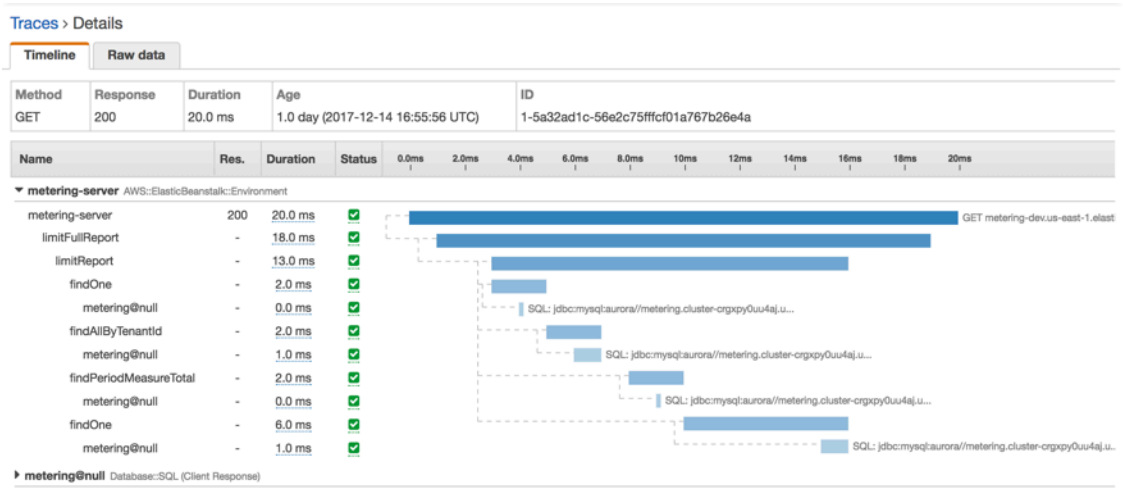
Der folgende Code erweitert die abstrakte Klasse`BaseAbstractXRayInterceptor`.

```
@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() {}

}
```

Der folgenden Code ist eine Klasse, die von X-Ray instrumentiert wird.

```
@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());
        }
    }
}
```

Wenn Sie Ihre Anwendung ordnungsgemäß konfiguriert haben, sollten Sie den vollständigen Aufruf-Stack der Anwendung sehen, vom Controller bis zu den Service-Aufrufe, wie im folgenden Screenshot der Konsole gezeigt.



Arbeiten mit Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Es gibt zwei Möglichkeiten, Ihre Anwendung Node.js so zu instrumentieren, dass sie Traces an X-Ray sendet:

- AWS Distro for OpenTelemetry JavaScript — Eine AWS Distribution, die eine Reihe von Open-Source-Bibliotheken zum Senden korrelierter Metriken und Traces über AWS Distro for Collector an mehrere AWS Überwachungslösungen CloudWatch AWS X-Ray, darunter Amazon und Amazon OpenSearch Service, bereitstellt. OpenTelemetry
- AWS X-Ray SDK für Node.js — Eine Reihe von Bibliotheken zum Generieren und Senden von Traces an X-Ray über den X-Ray-Daemon.

Weitere Informationen finden Sie unter Wahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs.

AWS Distro für OpenTelemetry JavaScript

Mit AWS Distro for OpenTelemetry (ADOT) JavaScript können Sie Ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Monitoring-Lösungen wie Amazon und Amazon CloudWatch Service AWS X-Ray senden. OpenSearch Die Verwendung von X-Ray mit AWS Distro for OpenTelemetry erfordert zwei Komponenten: ein OpenTelemetry SDK, das für die Verwendung mit X-Ray aktiviert ist, und das AWS Distro for OpenTelemetry Collector, das für die Verwendung mit X-Ray aktiviert ist.

Informationen zu den ersten Schritten finden Sie in der Dokumentation zur AWS Distribution. OpenTelemetry JavaScript

> **Note**
>
> ADOT JavaScript wird für alle serverseitigen Node.js Anwendungen unterstützt. ADOT JavaScript ist nicht in der Lage, Daten von Browser-Clients nach X-Ray zu exportieren.

Weitere Informationen zur Verwendung von AWS Distro for OpenTelemetry with AWS X-Ray und anderen AWS-Services finden Sie unter AWS Distro for OpenTelemetry oder The AWS Distro for Documentation. OpenTelemetry

Weitere Informationen zur Sprachunterstützung und -verwendung finden Sie unter AWS Observability on. GitHub

AWS X-Ray-SDK für Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK für Node.js ist eine Bibliothek für Express-Webanwendungen und Node.js Lambda-Funktionen, die Klassen und Methoden zum Generieren und Senden von Trace-Daten an den X-Ray-Daemon bereitstellt. Zu den Trace-Daten gehören Informationen über eingehende HTTP-Anfragen, die von der Anwendung bedient werden, sowie über Aufrufe, die die Anwendung mithilfe des AWS SDK oder der HTTP-Clients an nachgeschaltete Dienste sendet.

> **Note**
>
> Das X-Ray-SDK für Node.js ist ein Open-Source-Projekt, das für Node.js Versionen 14.x und höher unterstützt wird. Du kannst das Projekt verfolgen und Issues und Pull-Requests einreichen auf GitHub: github. com/aws/aws- xray-sdk-node

Wenn Sie Express nutzen, beginnen Sie, indem Sie das SDK als Middleware auf Ihrem Anwendungsserver hinzufügen, um eingehende Anforderungen zu verfolgen. Der Middleware erstellt für jede verfolgte Anforderung ein Segment und vervollständigt das Segment, nachdem die Antwort gesendet wurde. Während das Segment geöffnet ist, können Sie die SDK-Client-Methoden nutzen, um dem Segment Informationen hinzuzufügen, Untersegmente zu erstellen und nachgelagerte Aufrufe rückzuverfolgen. Das SDK erfasst auch automatisch Ausnahmen, die Ihre Anwendung ausgibt, während das Segment geöffnet ist.

Bei Lambda-Funktionen, die von einer instrumentierten Anwendung oder einem Dienst aufgerufen werden, liest Lambda den Tracing-Header und verfolgt automatisch Sampling-Anfragen. Für andere Funktionen können Sie Lambda so konfigurieren, dass eingehende Anfragen abgefragt und verfolgt werden. In beiden Fällen erstellt Lambda das Segment und stellt es dem X-Ray SDK zur Verfügung.

> **Note**
>
> Auf Lambda ist das X-Ray SDK optional. Wenn Sie es nicht in Ihrer Funktion verwenden, enthält Ihre Service-Map immer noch einen Knoten für den Lambda-Service und einen für jede Lambda-Funktion. Durch Hinzufügen des SDK können Sie Ihren Funktionscode instrumentieren, um Untersegmente zu dem von Lambda aufgezeichneten Funktionssegment hinzuzufügen. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Verwenden Sie als Nächstes das X-Ray-SDK für Node.js, um Ihr AWS SDK für die Clients JavaScript in Node.js zu instrumentieren. Immer wenn Sie einen Downstream AWS-Service oder eine Ressource mit einem instrumentierten Client aufrufen, zeichnet das SDK Informationen über den Anruf in einem Untersegment auf. AWS-Services und die Ressourcen, auf die Sie innerhalb der Services zugreifen, werden in der Trace-Map als Downstream-Knoten angezeigt, sodass Sie Fehler und Drosselungsprobleme bei einzelnen Verbindungen leichter identifizieren können.

Das X-Ray-SDK für Node.js bietet auch Instrumentierung für Downstream-Aufrufe von HTTP-Web APIs - und SQL-Abfragen. Umhüllen Sie den HTTP-Client in der SDK-Erfassungsmethode, um

Informationen zu ausgehenden HTTP-Anforderungen aufzuzeichnen. Für SQL-Clients verwenden Sie die Capture-Methode für Ihre Datenbank.

Die Middleware wendet Samplingregeln auf eingehende Anforderungen an, um zu ermitteln, welche Anforderungen rückverfolgt werden. Sie können das X-Ray SDK für Node.js konfigurieren, um das Sampling-Verhalten anzupassen oder Informationen über die AWS Rechenressourcen aufzuzeichnen, auf denen Ihre Anwendung ausgeführt wird.

Zeichnen Sie zusätzliche Informationen zu Anforderungen und den Aufgaben, die Ihre Anwendung ausführt, in Anmerkungen und Metadaten auf. Anmerkungen sind einfache Schlüsselwertpaare, die für die Verwendung mit Filterausdrücken indiziert werden, damit Sie nach Ablaufverfolgen mit bestimmten Daten suchen können. Metadateneinträge sind weniger einschränkend und können ganze Objekte und Arrays aufzeichnen – alle Daten, die in eine JSON zusammengefasst werden können.

> **Anmerkungen und Metadaten**
>
> Anmerkungen und Metadaten sind beliebiger Text, den Sie Segmenten mit dem X-Ray SDK hinzufügen. Anmerkungen werden für die Verwendung mit Filterausdrücken indexiert. Metadaten werden nicht indexiert, können aber im Rohsegment mit der X-Ray-Konsole oder API angezeigt werden. Jeder, dem Sie Lesezugriff auf X-Ray gewähren, kann diese Daten einsehen.

Wenn Sie viele instrumentierten Clients in Ihrem Code haben, kann ein einzelnes Anforderungssegmente viele Untersegmente enthalten, eines für jeden Aufruf mit einem instrumentierten Client. Sie können Untersegmente organisieren und gruppieren, indem Sie Client-Aufrufe in benutzerdefinierten Untersegmenten zusammenfassen. Sie können ein benutzerdefiniertes Untersegment für eine ganze Funktion oder eine Code-Abschnitt erstellen und Metadaten und Anmerkungen im Untersegment festhalten, anstatt alles im übergeordneten Segment aufzuzeichnen.

Referenzdokumentation zu den Klassen und Methoden des SDK finden Sie in der API-Referenz zum AWS X-Ray SDK for Node.js.

Voraussetzungen

Das X-Ray SDK für Node.js benötigt Node.js und die folgenden Bibliotheken:

- `atomic-batcher`— 1.0.2

- `cls-hooked`— 4.2.2
- `pkginfo`— 0,4,0
- `semver`— 5,3,0

Das SDK zieht diese Bibliotheken bei der Installation in NPM ein.

Um AWS SDK-Clients verfolgen zu können, benötigt das X-Ray-SDK für Node.js eine Mindestversion des AWS SDK für JavaScript in Node.js.

- `aws-sdk`— 2.7.15

Abhängigkeitsmanagement

Das X-Ray-SDK für Node.js ist bei NPM erhältlich.

- Package — `aws-xray-sdk`

Installieren Sie das SDK für eine lokale Bereitstellung in Ihrem Projektverzeichnis mit 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
```

Verwenden Sie die `--save`-Option zum Speichern von SDK in Abhängigkeit von `package.json` in Ihrer Anwendung.

```
~/nodejs-xray$ npm install aws-xray-sdk --save
aws-xray-sdk@3.3.3
```

Wenn Ihre Anwendung Abhängigkeiten hat, deren Versionen mit den Abhängigkeiten des X-Ray-SDK in Konflikt stehen, werden beide Versionen installiert, um die Kompatibilität sicherzustellen. Weitere Informationen finden Sie in der offiziellen NPM-Dokumentation zur Auflösung von Abhängigkeiten.

Node.js-Beispiele

Verwenden Sie das AWS X-Ray SDK für Node.js, um sich einen end-to-end Überblick über die Anfragen zu verschaffen, die durch Ihre Node.js -Anwendungen übertragen werden.

- Node.js Beispielanwendung aktiviert GitHub.

Konfiguration des X-Ray-SDK für Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können das X-Ray-SDK für Node.js mit Plug-ins so konfigurieren, dass es Informationen über den Dienst enthält, auf dem Ihre Anwendung ausgeführt wird, das standardmäßige Sampling-Verhalten ändern oder Sampling-Regeln hinzufügen, die für Anfragen an bestimmte Pfade gelten.

Sections

- Service-Plugins
- Regeln für die Probenahme
- Protokollierung
- Adresse des X-Ray-Daemons

- Umgebungsvariablen

Service-Plugins

Wird verwendet`plugins`, um Informationen über den Dienst aufzuzeichnen, der Ihre Anwendung hostet.

Plugins

- Amazon EC2 — `EC2Plugin` fügt die Instance-ID, die Availability Zone und die CloudWatch Logs-Gruppe hinzu.
- Elastic Beanstalk — `ElasticBeanstalkPlugin` fügt den Umgebungsnamen, die Versionsbezeichnung und die Bereitstellungs-ID hinzu.
- Amazon ECS — `ECSPlugin` fügt die Container-ID hinzu.

Um ein Plug-in zu verwenden, konfigurieren Sie das X-Ray SDK für den Client Node.js mithilfe der `config` Methode.

Example app.js – Plugins

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.config([AWSXRay.plugins.EC2Plugin,AWSXRay.plugins.ElasticBeanstalkPlugin]);
```

Das SDK verwendet auch Plugin-Einstellungen, um das `origin` Feld für das Segment festzulegen. Dies gibt den AWS Ressourcentyp an, auf dem Ihre Anwendung ausgeführt wird. Wenn Sie mehrere Plugins verwenden, verwendet das SDK die folgende Auflösungsreihenfolge, um den Ursprung zu bestimmen: ElasticBeanstalk > EKS > ECS > EC2.

Regeln für die Probenahme

Das SDK verwendet die Sampling-Regeln, die Sie in der X-Ray-Konsole definieren, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen. Die Standardregel verfolgt die erste Anfrage jede Sekunde und fünf Prozent aller weiteren Anfragen aller Dienste, die Traces an X-Ray senden. Erstellen Sie zusätzliche Regeln in der X-Ray-Konsole, um die Menge der aufgezeichneten Daten für jede Ihrer Anwendungen anzupassen.

Das SDK wendet benutzerdefinierte Regeln in der Reihenfolge an, in der sie definiert sind. Wenn eine Anfrage mehreren benutzerdefinierten Regeln entspricht, wendet das SDK nur die erste Regel an.

> **Note**
>
> Wenn das SDK X-Ray nicht erreichen kann, um Sampling-Regeln abzurufen, kehrt es zu einer lokalen Standardregel zurück, die die erste Anfrage pro Sekunde und fünf Prozent aller zusätzlichen Anfragen pro Host vorsieht. Dies kann passieren, wenn der Host nicht berechtigt ist APIs, Sampling aufzurufen, oder wenn er keine Verbindung zum X-Ray-Daemon herstellen kann, der als TCP-Proxy für API-Aufrufe durch das SDK fungiert.

Sie können das SDK auch so konfigurieren, dass Sampling-Regeln aus einem JSON-Dokument geladen werden. Das SDK kann lokale Regeln als Backup für Fälle verwenden, in denen X-Ray Sampling nicht verfügbar ist, oder ausschließlich lokale Regeln verwenden.

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
  }
}
```

In diesem Beispiel werden eine benutzerdefinierte Regel und eine Standardregel definiert. Die benutzerdefinierte Regel wendet eine Stichprobenrate von fünf Prozent an, ohne dass eine Mindestanzahl von Anfragen für Pfade verfolgt werden muss. `/api/move/` Die Standardregel verfolgt die erste Anfrage jede Sekunde und 10 Prozent der weiteren Anfragen.

Der Nachteil der lokalen Definition von Regeln besteht darin, dass das feste Ziel von jeder Instanz des Rekorders unabhängig angewendet wird, anstatt vom X-Ray-Dienst verwaltet zu werden. Wenn

Sie mehr Hosts bereitstellen, wird die feste Rate vervielfacht, wodurch es schwieriger wird, die Menge der aufgezeichneten Daten zu kontrollieren.

Wenn aktiviert AWS Lambda, können Sie die Samplerate nicht ändern. Wenn Ihre Funktion von einem instrumentierten Dienst aufgerufen wird, werden Aufrufe, die Anfragen generierten, die von diesem Dienst abgetastet wurden, von Lambda aufgezeichnet. Wenn aktives Tracing aktiviert ist und kein Tracing-Header vorhanden ist, trifft Lambda die Stichprobenentscheidung.

Um Backup-Regeln zu konfigurieren, weisen Sie das X-Ray SDK for Node.js an, Sampling-Regeln aus einer Datei mit zu ladensetSamplingRules.

Example app.js – Sampling-Regeln aus einer Datei

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
```

Sie können Ihre Regeln auch in Code definieren und als Objekt an `setSamplingRules` übergeben.

Example app.js – Sampling-Regeln aus einem Objekt

```
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);
```

Um nur lokale Regeln zu verwenden, rufen Sie `disableCentralizedSampling` auf.

```
AWSXRay.middleware.disableCentralizedSampling()
```

Protokollierung

Zum Protokollieren der SDK-Ausgabe rufen Sie `AWSXRay.setLogger(logger)` auf, wobei `logger` ein Objekt ist, dass Standard-Protokollierungsmethoden (`warn`, `info`usw.) bereitstellt.

Standardmäßig protokolliert das SDK Fehlermeldungen auf der Konsole mithilfe der Standardmethoden für das Konsolenobjekt. Die Protokollebene des integrierten Loggers kann

entweder mithilfe der `AWS_XRAY_LOG_LEVEL` Umgebungsvariablen `AWS_XRAY_DEBUG_MODE` oder festgelegt werden. Eine Liste der gültigen Werte auf Protokollebene finden Sie unter Umgebungsvariablen.

Wenn Sie ein anderes Format oder ein anderes Ziel für die Protokolle angeben möchten, können Sie dem SDK Ihre eigene Implementierung der Logger-Schnittstelle zur Verfügung stellen, wie unten gezeigt. Jedes Objekt, das diese Schnittstelle implementiert, kann verwendet werden. Das bedeutet, dass viele Logging-Bibliotheken, z. B. Winston, verwendet und direkt an das SDK übergeben werden könnten.

Example app.js – Protokollierung

```
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]);
```

Rufen Sie vor der Ausführung anderer Konfigurationsmethoden `setLogger` auf, um sicherzustellen, dass Sie die Ausgabe dieser Vorgänge erfassen.

Adresse des X-Ray-Daemons

Wenn der X-Ray-Daemon auf einem anderen Port oder Host lauscht als`127.0.0.1:2000`, können Sie das X-Ray-SDK für Node.js so konfigurieren, dass Trace-Daten an eine andere Adresse gesendet werden.

```
AWSXRay.setDaemonAddress('host:port');
```

Sie können den Host anhand des Namens oder der Adresse angeben. IPv4

Example app.js – Daemon-Adresse

```
var AWSXRay = require('aws-xray-sdk');
```

```
AWSXRay.setDaemonAddress('daemonhost:8082');
```

Wenn Sie den Daemon so konfiguriert haben, dass er auf verschiedenen Ports für TCP und UDP wartet, können Sie beides in den Einstellungen für die Daemon-Adresse angeben.

Example app.js – Daemon-Adresse auf separaten Ports

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('tcp:daemonhost:8082 udp:daemonhost:8083');
```

Sie können die Daemon-Adresse auch festlegen, indem Sie die `AWS_XRAY_DAEMON_ADDRESS` Umgebungsvariable verwenden.

Umgebungsvariablen

Sie können Umgebungsvariablen verwenden, um das X-Ray SDK für Node.js zu konfigurieren. Das SDK unterstützt die folgenden Variablen.

- `AWS_XRAY_CONTEXT_MISSING`— Legt fest, `RUNTIME_ERROR` dass Ausnahmen ausgelöst werden, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, obwohl kein Segment geöffnet ist.

  Zulässige Werte

  - `RUNTIME_ERROR`— Löst eine Laufzeitausnahme aus.
  - `LOG_ERROR`— Fehler protokollieren und fortfahren (Standard).
  - `IGNORE_ERROR`— Fehler ignorieren und fortfahren.

  Fehler im Zusammenhang mit fehlenden Segmenten oder Untersegmenten können auftreten, wenn Sie versuchen, einen instrumentierten Client in Startcode zu verwenden, der ausgeführt wird, wenn keine Anfrage geöffnet ist, oder in Code, der einen neuen Thread erzeugt.
- `AWS_XRAY_DAEMON_ADDRESS`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest. Standardmäßig verwendet `127.0.0.1:2000` das SDK sowohl Trace-Daten (UDP) als auch Sampling-Daten (TCP). Verwenden Sie diese Variable, wenn Sie den Daemon so konfiguriert haben, dass er auf einem anderen Port lauscht oder wenn er auf einem anderen Host läuft.

  Format

  - Derselbe Port — *`address`*`:`*`port`*

  - Verschiedene Anschlüsse — `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_DEBUG_MODE`— Stellen Sie diese Option ein`TRUE`, um das SDK so zu konfigurieren, dass es Protokolle auf `debug` Ebene 2 ausgibt.
- `AWS_XRAY_LOG_LEVEL` — Legt eine Protokollebene für den Standard-Logger fest. Gültige Werte sind `debug`, `info`, `warn`, `error` und `silent`. Dieser Wert wird ignoriert, wenn er auf gesetzt AWS_XRAY_DEBUG_MODE ist`TRUE`.
- `AWS_XRAY_TRACING_NAME`— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet. Überschreibt den für die Express-Middleware festgelegten Segmentnamen.

Nachverfolgung eingehender Anfragen mit dem X-Ray SDK für Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können das X-Ray SDK für Node.js verwenden, um eingehende HTTP-Anfragen zu verfolgen, die Ihre Express- und Restify-Anwendungen auf einer EC2 Instance in Amazon EC2 oder Amazon AWS Elastic Beanstalk ECS bearbeiten.

Das X-Ray SDK für Node.js bietet Middleware für Anwendungen, die die Express- und Restify-Frameworks verwenden. Wenn Sie die X-Ray-Middleware zu Ihrer Anwendung hinzufügen, erstellt das X-Ray-SDK für Node.js ein Segment für jede gesampelte Anfrage. Dieses Segment umfasst Dauer, Methode und Status der HTTP-Anforderung. Die zusätzliche Instrumentierung schafft Untersegmente zu diesem Segment.

> **Note**
>
> Für AWS Lambda Funktionen erstellt Lambda für jede abgetastete Anfrage ein Segment. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Jedes Segment hat einen Namen, der Ihre Anwendung in der Service Map identifiziert. Das Segment kann statisch benannt werden, oder Sie können das SDK so konfigurieren, dass es dynamisch auf der Grundlage des Host-Headers in der eingehenden Anfrage benannt wird. Mit der dynamischen Benennung können Sie Traces auf der Grundlage des Domainnamens in der Anfrage gruppieren und einen Standardnamen anwenden, wenn der Name nicht einem erwarteten Muster entspricht (z. B. wenn der Host-Header gefälscht ist).

> **Weitergeleitete Anfragen**
>
> Wenn ein Load Balancer oder ein anderer Vermittler eine Anfrage an Ihre Anwendung weiterleitet, nimmt X-Ray die Client-IP aus dem `X-Forwarded-For` Header in der Anfrage und nicht aus der Quell-IP im IP-Paket. Die Client-IP, die für eine weitergeleitete Anfrage aufgezeichnet wird, kann gefälscht sein und sollte daher nicht als vertrauenswürdig eingestuft werden.

Wenn eine Anfrage weitergeleitet wird, legt das SDK ein zusätzliches Feld im Segment fest, um dies anzuzeigen. Wenn das Segment das Feld enthält, das auf `x_forwarded_for` gesetzt ist`true`, wurde die Client-IP aus dem `X-Forwarded-For` Header in der HTTP-Anfrage übernommen.

Der Meldungshandler erzeugt für jede eingehende Anforderung ein Segment mit einem `http`-Block, der die folgenden Informationen enthält:

- HTTP-Methode — GET, POST, PUT, DELETE usw.
- Client-Adresse — Die IP-Adresse des Clients, der die Anfrage gesendet hat.
- Antwortcode — Der HTTP-Antwortcode für die abgeschlossene Anfrage.
- Timing — Die Startzeit (als die Anfrage empfangen wurde) und die Endzeit (als die Antwort gesendet wurde).
- Benutzeragent — Der `user-agent` aus der Anfrage.
- Länge des Inhalts — Die Länge `content-length` der Antwort.

Sections

- Nachverfolgung eingehender Anforderungen mit Express
- Nachverfolgung eingehender Anforderungen mit Restify
- Konfiguration einer Segmentbenennungsstrategie

Nachverfolgung eingehender Anforderungen mit Express

Zum Verwenden der Express-Middleware initialisieren Sie den SDK-Client und nutzen Sie die Middleware, die von der `express.openSegment`-Funktion zurückgegeben wird, bevor Sie Ihre Routen festlegen.

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());
```

Nachdem Sie Ihre Routen definiert haben, verwenden Sie die Ausgabe von `express.closeSegment` wie abgebildet, um alle Fehler zu behandeln, die vom X-Ray SDK für Node.js zurückgegeben werden.

Nachverfolgung eingehender Anforderungen mit Restify

Um die Restify-Middleware zu verwenden, initialisieren Sie den SDK-Client und führen Sie `enable` aus. Übergeben Sie den Namen Ihres Restify-Servers und des Segments.

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');
});
```

Konfiguration einer Segmentbenennungsstrategie

AWS X-Ray verwendet einen Dienstnamen, um Ihre Anwendung zu identifizieren und sie von den anderen Anwendungen, Datenbanken, externen AWS Ressourcen und Ressourcen zu unterscheiden APIs, die Ihre Anwendung verwendet. Wenn das X-Ray SDK Segmente für eingehende Anfragen generiert, zeichnet es den Dienstnamen Ihrer Anwendung im Namensfeld des Segments auf.

Das X-Ray SDK kann Segmente nach dem Hostnamen im HTTP-Anforderungsheader benennen. Dieser Header kann jedoch gefälscht sein, was zu unerwarteten Knoten in Ihrer Service Map führen kann. Um zu verhindern, dass das SDK Segmente aufgrund von Anfragen mit gefälschten Host-Headern falsch benennt, müssen Sie einen Standardnamen für eingehende Anfragen angeben.

Wenn Ihre Anwendung Anfragen für mehrere Domänen bearbeitet, können Sie das SDK so konfigurieren, dass es eine dynamische Benennungsstrategie verwendet, um dies in Segmentnamen widerzuspiegeln. Eine dynamische Benennungsstrategie ermöglicht es dem SDK, den Hostnamen für Anfragen zu verwenden, die einem erwarteten Muster entsprechen, und den Standardnamen auf Anfragen anzuwenden, bei denen dies nicht der Fall ist.

Beispielsweise könnten Sie eine einzige Anwendung haben, die Anfragen an drei Subdomänen — `www.example.com`api.example.com, und — bedient. `static.example.com` Sie können eine dynamische Benennungsstrategie mit dem Muster verwenden`*.example.com`, um Segmente für jede Subdomain mit einem anderen Namen zu identifizieren, was zu drei Dienstknoten auf der Service-Map führt. Wenn Ihre Anwendung Anfragen mit einem Hostnamen empfängt, der nicht dem Muster entspricht, wird auf der Service Map ein vierter Knoten mit einem von Ihnen angegebenen Fallback-Namen angezeigt.

Wenn Sie denselben Namen für alle Segmente verwenden möchten, geben Sie bei der Initialisierung der Middleware den Namen Ihrer Anwendung, wie in den vorherigen Abschnitten gezeigt, ein.

> **Note**
>
> Sie können den mit der `AWS_XRAY_TRACING_NAME`-Umgebungsvariablen in Code definierten standardmäßigen Dienstnamen überschreiben.

Eine dynamische Benennungsstrategie definiert ein Muster, dem Hostnamen entsprechen sollten, sowie einen Standardnamen, der verwendet wird, wenn der Hostname in der HTTP-Anforderung nicht mit diesem Muster übereinstimmt. Zur dynamischen Segmentbenennung verwenden Sie `AWSXRay.middleware.enableDynamicNaming`.

Example app.js – dynamische Segmentnamen

Wenn der Hostname in der Anforderung dem Muster `*.example.com` entspricht, verwenden Sie den Hostnamen. Verwenden Sie andernfalls `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());
```

Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK für Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Aufrufe tätigt, um Daten AWS-Services zu speichern, in eine Warteschlange zu schreiben oder Benachrichtigungen zu senden, verfolgt das X-Ray SDK for Node.js die Aufrufe im Downstream in Untersegmenten. Traced und Ressourcen AWS-Services, auf die Sie innerhalb dieser Services zugreifen (z. B. ein Amazon S3-Bucket oder eine Amazon SQS SQS-Warteschlange), werden als Downstream-Knoten auf der Trace-Map in der X-Ray-Konsole angezeigt.

AWS Instrumenten-SDK-Clients, die Sie über AWS SDK für JavaScript V2 oder AWS SDK für JavaScript V3 erstellen. Jede AWS SDK-Version bietet unterschiedliche Methoden zur Instrumentierung von AWS SDK-Clients.

> **Note**
>
> Derzeit gibt das AWS X-Ray SDK für Node.js bei der Instrumentierung von AWS SDK für JavaScript V3-Clients weniger Segmentinformationen zurück als bei der Instrumentierung von V2-Clients. Beispielsweise geben Untersegmente, die Aufrufe von DynamoDB darstellen, den Tabellennamen nicht zurück. Wenn Sie diese Segmentinformationen in Ihren Traces benötigen, sollten Sie die Verwendung von V2 in Betracht ziehen. AWS SDK für JavaScript

AWS SDK für JavaScript V2

Sie können alle AWS SDK V2-Clients instrumentieren, indem Sie Ihre `aws-sdk` require-Anweisung in einen Call to einbinden`AWSXRay.captureAWS`.

Example app.js — AWS SDK-Instrumentierung

```
const AWS = AWSXRay.captureAWS(require('aws-sdk'));
```

Um einzelne Clients zu instrumentieren, binden Sie Ihren AWS SDK-Client in einen Aufruf von ein`AWSXRay.captureAWSClient`. Instrumentieren Sie beispielsweise einen `AmazonDynamoDB`-Client wie folgt:

Example app.js — DynamoDB-Client-Instrumentierung

```
    const AWSXRay = require('aws-xray-sdk');
...
    const ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
```

> **Warning**
>
> Verwenden Sie nicht `captureAWS` und `captureAWSClient` zusammen. Dies führt zu doppelten Untersegmenten.

Wenn Sie TypeScriptmit ECMAScriptModulen (ESM) Ihren JavaScript Code laden möchten, verwenden Sie das folgende Beispiel, um Bibliotheken zu importieren:

Example app.js — AWS SDK-Instrumentierung

```
import * as AWS from 'aws-sdk';
```

```
import * as AWSXRay from 'aws-xray-sdk';
```

Verwenden Sie den folgenden Code, um alle AWS Clients mit ESM zu instrumentieren:

Example app.js — AWS SDK-Instrumentierung

```
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();
```

Für alle Dienste können Sie den Namen der aufgerufenen API in der X-Ray-Konsole sehen. Für eine Untergruppe von Diensten fügt das X-Ray SDK dem Segment Informationen hinzu, um die Service Map detaillierter zu gestalten.

Wenn Sie beispielsweise einen Aufruf mit einem instrumentierten DynamoDB-Client tätigen, fügt das SDK den Tabellennamen dem Segment für Aufrufe hinzu, die auf eine Tabelle abzielen. In der Konsole wird jede Tabelle als separater Knoten in der Service Map angezeigt, mit einem generischen DynamoDB-Knoten für Aufrufe, die nicht auf eine Tabelle abzielen.

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements

```
{
  "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",
  }
}
```

Wenn Sie auf benannte Ressourcen zugreifen, werden durch Aufrufe der folgenden Services weitere Knoten in der Service-Übersicht erstellt. Durch Aufrufe, die keinen bestimmten Ressourcen gelten, wird ein generischer Knoten für den Service erstellt.

- Amazon DynamoDB — Tabellenname
- Amazon Simple Storage Service — Bucket und Schlüsselname
- Amazon Simple Queue Service — Name der Warteschlange

AWS SDK für JavaScript V3

Die AWS SDK für JavaScript Version 3 ist modular aufgebaut, sodass Ihr Code nur die Module lädt, die er benötigt. Aus diesem Grund ist es nicht möglich, alle AWS SDK-Clients zu instrumentieren, da V3 die `captureAWS` Methode nicht unterstützt.

Wenn Sie TypeScript mit ECMAScript Modules (ESM) Ihren JavaScript Code laden möchten, können Sie das folgende Beispiel verwenden, um Bibliotheken zu importieren:

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

Instrumentieren Sie jeden AWS SDK-Client mit der `AWSXRay.captureAWSv3Client` Methode. Instrumentieren Sie beispielsweise einen `AmazonDynamoDB`-Client wie folgt:

Example app.js — DynamoDB-Client-Instrumentierung mit SDK für Javascript V3

```
    const AWSXRay = require('aws-xray-sdk');
    const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
...
    const ddb = AWSXRay.captureAWSv3Client(new DynamoDBClient({ region:
 "region" }));
```

Bei Verwendung von AWS SDK für JavaScript V3 werden Metadaten wie Tabellenname, Bucket- und Schlüsselname oder Warteschlangenname derzeit nicht zurückgegeben. Daher enthält die Trace-Map keine separaten Knoten für jede benannte Ressource, wie dies bei der Instrumentierung von AWS SDK-Clients mit V2 der Fall wäre. AWS SDK für JavaScript

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements bei Verwendung von V3 AWS SDK für 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",
  }
}
```

Verfolgen von Aufrufen an Downstream-HTTP-Webservices mithilfe des X-Ray SDK für Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Microservices oder öffentliches HTTP aufruft APIs, können Sie den X-Ray SDK for Node.js Client verwenden, um diese Aufrufe zu instrumentieren und die API als Downstream-Service zum Service Graph hinzuzufügen.

Übergeben Sie Ihren `http` oder `https` Client an die `captureHTTPs` Methode X-Ray SDK for Node.js, um ausgehende Anrufe zu verfolgen.

> **Note**
>
> Aufrufe, die HTTP-Anforderungsbibliotheken von Drittanbietern wie Axios oder Superagent verwenden, werden durch die `captureHTTPsGlobal()`-API unterstützt und werden weiterhin nachverfolgt, wenn sie das `http`-Modul verwenden.

Example app.js – HTTP-Client

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Zur Aktivierung der Nachverfolgung auf allen HTTP-Clients rufen Sie `captureHTTPsGlobal` auf, bevor Sie `http` laden.

Example app.js – HTTP-Client (Global)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

Wenn Sie einen Aufruf einer Downstream-Web-API instrumentieren, zeichnet das X-Ray-SDK für Node.js ein Untersegment auf, das Informationen über die HTTP-Anfrage und -Antwort enthält. X-Ray verwendet das Untersegment, um ein abgeleitetes Segment für die Remote-API zu generieren.

Example Untersegment für einen nachgelagerten HTTP-Aufruf

```
{
  "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 Abgeleitetes Segment für einen nachgelagerten HTTP-Anruf

```
{
  "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
}
```

Verfolgen von SQL-Abfragen mit dem X-Ray SDK für Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Instrumentieren Sie SQL-Datenbankabfragen, indem Sie Ihren SQL-Client in die entsprechende Client-Methode des X-Ray SDK for Node.js einbinden.

- 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);
```

Wenn Sie mit einem instrumentierten Client SQL-Abfragen vornehmen, zeichnet das X-Ray SDK for Node.js in einem Untersegment Informationen über die Verbindung und die Abfrage auf.

Zusätzliche Daten in SQL-Untersegmenten einbeziehen

Sie können Untersegmenten, die für SQL-Abfragen generiert wurden, zusätzliche Informationen hinzufügen, sofern diese einem SQL-Feld auf der Zulassungsliste zugeordnet sind. Um beispielsweise die bereinigte SQL-Abfragezeichenfolge in einem Untersegment aufzuzeichnen, können Sie sie direkt zum SQL-Objekt des Untersegments hinzufügen.

Example Weisen Sie SQL einem Untersegment zu

```
    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;
}
```

Eine vollständige Liste der SQL-Felder auf der Zulassungsliste finden Sie unter SQL-Abfragen im AWS X-Ray Entwicklerhandbuch.

Generieren benutzerdefinierter Untersegmente mit dem X-Ray SDK für Node.js

Untersegmente erweitern das Segment eines Traces um Details über die Arbeit, die zur Bearbeitung einer Anfrage geleistet wurde. Jedes Mal, wenn Sie einen Anruf mit einem instrumentierten Client tätigen, zeichnet das X-Ray-SDK die in einem Untersegment generierten Informationen auf. Sie können zusätzliche Untersegmente erstellen, um andere Untersegmente zu gruppieren, die Leistung eines Codeabschnitts zu messen oder Anmerkungen und Metadaten aufzuzeichnen.

Benutzerdefinierte Express-Untersegmente

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Um ein benutzerdefiniertes Untersegment für eine Funktion zu erstellen, die Aufrufe des nachgelagerten Service vornimmt, verwenden Sie die `captureAsyncFunc`-Funktion.

Example app.js – benutzerdefinierte Untersegmente 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();
};
```

In diesem Beispiel erstellt die Anwendung ein benutzerdefiniertes Untersegment namens `send` für Aufrufe der `sendRequest`-Funktion. `captureAsyncFunc` übergibt ein Untersegment, das Sie innerhalb der Callback-Funktion schließen müssen, wenn die asynchronen Aufrufe der Funktion abgeschlossen sind.

Für synchrone Funktionen können Sie die `captureFunc`-Funktion verwenden, mit der das Untersegment automatisch geschlossen wird, sobald der Funktionsblock ausgeführt wurde.

Wenn Sie ein Untersegment innerhalb eines Segments oder eines anderen Untersegments erstellen, generiert das X-Ray SDK for Node.js eine ID dafür und zeichnet die Start- und Endzeit auf.

Example Untersegment mit Metadaten

```
"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"
    }
  },
```

Benutzerdefinierte Lambda-Untersegmente

Das SDK ist so konfiguriert, dass es automatisch ein Platzhalter-Fassadensegment erstellt, wenn es erkennt, dass es in Lambda ausgeführt wird. Um ein grundlegendes Untersegment zu erstellen, das einen einzelnen `AWS::Lambda::Function` Knoten auf der X-Ray-Trace-Map erstellt, rufen Sie das Fassadensegment auf und verwenden es für neue Zwecke. Wenn Sie manuell ein neues Segment mit einer neuen ID erstellen (während Sie die Ablaufverfolgung-ID, die übergeordnete ID und die Sampling-Entscheidung teilen), können Sie ein neues Segment senden.

Example app.js – benutzerdefinierte manuelle Untersegmente

```
const segment = AWSXRay.getSegment(); //returns the facade segment
const subsegment = segment.addNewSubsegment('subseg');
...
subsegment.close();
//the segment is closed by the SDK automatically
```

Hinzufügen von Anmerkungen und Metadaten zu Segmenten mit dem X-Ray SDK for Node.js

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können zusätzliche Informationen über Anfragen, die Umgebung oder Ihre Anwendung mit Anmerkungen und Metadaten aufzeichnen. Sie können Anmerkungen und Metadaten zu den Segmenten hinzufügen, die das X-Ray SDK erstellt, oder zu benutzerdefinierten Untersegmenten, die Sie erstellen.

Anmerkungen sind Schlüssel-Wert-Paare mit Zeichenfolgen-, Zahlen- oder booleschen Werten. Anmerkungen sind für die Verwendung mit Filterausdrücken indexiert. Berücksichtigen Sie Anmerkungen, um Daten zur Gruppierung von Ablaufverfolgungen in der Konsole zu verwenden, oder wenn Sie die `GetTraceSummaries`-API aufrufen.

Metadaten sind Schlüssel-Wert-Paare, die Werte beliebigen Typs enthalten können, einschließlich Objekte und Listen, aber nicht für die Verwendung mit Filterausdrücken indexiert sind. Verwenden Sie Metadaten, um zusätzliche Daten aufzuzeichnen, die Sie im Trace speichern möchten, aber nicht für die Suche verwenden müssen.

Zusätzlich zu Anmerkungen und Metadaten können Sie auch Benutzer-ID-Zeichenfolgen in Segmenten aufzeichnen. Benutzer IDs werden in einem separaten Feld in Segmenten aufgezeichnet und für die Verwendung bei der Suche indexiert.

Sections

- Anmerkungen mit dem X-Ray SDK für Node.js aufnehmen
- Metadaten mit dem X-Ray SDK für Node.js aufzeichnen
- Benutzer IDs mit dem X-Ray SDK für Node.js aufzeichnen

Anmerkungen mit dem X-Ray SDK für Node.js aufnehmen

Verwenden Sie Anmerkungen, um Informationen zu Segmenten oder Untersegmenten, die zur Suche indiziert werden sollten, aufzuzeichnen.

Anmerkung zu Anforderungen

- Schlüssel — Der Schlüssel für eine X-Ray-Anmerkung kann bis zu 500 alphanumerische Zeichen enthalten. Sie können keine anderen Leerzeichen oder Symbole als einen Punkt oder Punkt (.) verwenden
- Werte — Der Wert für eine X-Ray-Anmerkung kann bis zu 1.000 Unicode-Zeichen enthalten.
- Die Anzahl der Anmerkungen — Sie können bis zu 50 Anmerkungen pro Spur verwenden.

So zeichnen Sie Anmerkungen auf

1. Erhalten Sie eine Referenz zum aktuellen Segment oder Untersegment.

```
var AWSXRay = require('aws-xray-sdk');
...
var document = AWSXRay.getSegment();
```

2. Rufen Sie `addAnnotation` mit einem Aktivierungsschlüssel und einem booleschen Wert oder einem Zeichenfolgenwert auf.

```
document.addAnnotation("mykey", "my value");
```

   Das folgende Beispiel zeigt, wie Sie `putAnnotation` mit einem String-Schlüssel aufrufen, der einen Punkt und einen booleschen Wert, eine Zahl oder einen String-Wert enthält.

```
document.putAnnotation("testkey.test", "my value");
```

Das SDK zeichnet Anmerkungen als Schlüssel-Wert-Paare in einem `annotations`-Objekt im Segmentdokument auf. Wenn `addAnnotation` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Nutzen Sie das `annotation[`*`key`*`]`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen durch Anmerkungen mit bestimmten Werten zu finden.

Example app.js – Anmerkungen

```
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) {
...
```

Metadaten mit dem X-Ray SDK für Node.js aufzeichnen

Verwenden Sie Metadaten, um Segment- oder Untersegmentinformationen aufzuzeichnen, die nicht zur Suche indiziert werden müssen. Metadatenwerte sind Zeichenfolgen, Zahlen, boolesche Werte oder andere Objekte, die in Form eines JSON-Objekts oder eines Arrays angeordnet sein können.

So zeichnen Sie Metadaten auf

1. Erhalten Sie eine Referenz zum aktuellen Segment oder Untersegment.

   ```
   var AWSXRay = require('aws-xray-sdk');
   ...
   var document = AWSXRay.getSegment();
   ```

2. Rufen Sie `addMetadata` mit einem Zeichenfolgenschlüssel, einem booleschen Wert, einer Zeichenfolge oder einem Objektwert und einem Zeichenfolgen-Namespace auf.

   ```
   document.addMetadata("my key", "my value", "my namespace");
   ```

   oder

   Rufen Sie `addMetadata` nur mit einem Aktivierungsschlüssel und einem Wert auf.

   ```
   document.addMetadata("my key", "my value");
   ```

Wenn Sie keinen Namespace angeben, verwendet SDK `default`. Wenn `addMetadata` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Benutzer IDs mit dem X-Ray SDK für Node.js aufzeichnen

Zeichnen Sie Segmente des Benutzers IDs auf Anfrage auf, um den Benutzer zu identifizieren, der die Anfrage gesendet hat. Dieser Vorgang ist nicht mit AWS Lambda Funktionen kompatibel, da Segmente in Lambda-Umgebungen unveränderlich sind. Der `setUser`-Aufruf kann nur auf Segmente angewendet werden, nicht auf Untersegmente.

Um einen Benutzer aufzuzeichnen IDs

1. Erhalten Sie eine Referenz zum aktuellen Segment oder Untersegment.

   ```
   var AWSXRay = require('aws-xray-sdk');
   ...
   var document = AWSXRay.getSegment();
   ```

2. Rufen Sie `setUser()` mit einer Zeichenfolgen-ID des Benutzers auf, der die Anforderung gesendet hat.

   ```
   var user = 'john123';

   AWSXRay.getSegment().setUser(user);
   ```

Sie können `setUser` aufrufen, um die Benutzer-ID aufzuzeichnen, sobald die Express-Anwendung mit der Verarbeitung einer Anfrage beginnt. Wenn Sie das Segment nur zur Einrichtung der Benutzer-ID verwenden, können Sie die Aufrufe in einer einzelnen Zeile anordnen.

Example app.js – Benutzer-ID

```
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) {
...
```

Nutzen Sie das `user`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen einer Benutzer-ID zu finden.

Verwenden von Python

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Es gibt zwei Möglichkeiten, Ihre Python-Anwendung so zu instrumentieren, dass sie Traces an X-Ray sendet:

- AWS Distro for OpenTelemetry Python — Eine AWS Distribution, die eine Reihe von Open-Source-Bibliotheken zum Senden korrelierter Metriken und Traces über AWS Distro for Collector an mehrere AWS Überwachungslösungen CloudWatch AWS X-Ray, darunter Amazon und Amazon OpenSearch Service, bereitstellt. OpenTelemetry
- AWS X-Ray SDK für Python — Eine Reihe von Bibliotheken zum Generieren und Senden von Traces an X-Ray über den X-Ray-Daemon.

Weitere Informationen finden Sie unter Wahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs.

AWS Distro für Python OpenTelemetry

Mit der AWS Distro for OpenTelemetry (ADOT) Python können Sie Ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Monitoring-Lösungen wie Amazon und Amazon CloudWatch Service AWS X-Ray senden. OpenSearch Die Verwendung von X-Ray mit ADOT erfordert zwei Komponenten: ein OpenTelemetry SDK, das für die Verwendung mit X-Ray aktiviert ist, und das AWS Distro for OpenTelemetry Collector, das für die Verwendung mit X-Ray aktiviert ist. ADOT Python bietet Unterstützung für automatische Instrumentierung, sodass Ihre Anwendung Traces ohne Codeänderungen senden kann.

Informationen zu den ersten Schritten finden Sie in der Dokumentation zu AWS Distro for OpenTelemetry Python.

Weitere Informationen zur Verwendung von AWS Distro for OpenTelemetry with AWS X-Ray und anderen AWS-Services finden Sie unter AWS Distro for OpenTelemetry oder Distro for Documentation AWS . OpenTelemetry

Weitere Informationen zur Sprachunterstützung und -verwendung finden Sie unter AWS Observability on. GitHub

AWS X-Ray SDK für Python

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK für Python ist eine Bibliothek für Python-Webanwendungen, die Klassen und Methoden zum Generieren und Senden von Trace-Daten an den X-Ray-Daemon bereitstellt. Zu den Trace-Daten gehören Informationen über eingehende HTTP-Anfragen, die von der Anwendung bedient werden, sowie über Aufrufe, die die Anwendung mithilfe des AWS SDK, der HTTP-Clients oder eines SQL-Datenbankkonnektors an Downstream-Dienste sendet. Sie können Segmente auch manuell erstellen und Debug-Informationen Anmerkungen und Metadaten hinzufügen.

Sie können das SDK mit `pip` herunterladen.

```
$ pip install aws-xray-sdk
```

> **Note**
>
> Das X-Ray SDK für Python ist ein Open-Source-Projekt. Du kannst das Projekt verfolgen und Issues und Pull-Requests einreichen auf GitHub: github. com/aws/aws- xray-sdk-python

Wenn Sie Django oder Flask nutzen, beginnen Sie, indem Sie die SDK-Middleware zu Ihrer Anwendung hinzufügen, um eingehende Anforderungen zu verfolgen. Der Middleware erstellt für jede verfolgte Anforderung ein Segment und vervollständigt das Segment, nachdem die Antwort gesendet wurde. Während das Segment geöffnet ist, können Sie die SDK-Client-Methoden nutzen, um dem Segment Informationen hinzuzufügen, Untersegmente zu erstellen und nachgelagerte Aufrufe rückzuverfolgen. Das SDK erfasst auch automatisch Ausnahmen, die Ihre Anwendung ausgibt, während das Segment geöffnet ist. Bei anderen Anwendungen können Sie Segmente manuell erstellen.

Bei Lambda-Funktionen, die von einer instrumentierten Anwendung oder einem Dienst aufgerufen werden, liest Lambda den Tracing-Header und verfolgt automatisch Sampling-Anfragen. Für andere Funktionen können Sie Lambda so konfigurieren, dass eingehende Anfragen abgefragt und verfolgt werden. In beiden Fällen erstellt Lambda das Segment und stellt es dem X-Ray SDK zur Verfügung.

> **Note**
>
> Auf Lambda ist das X-Ray SDK optional. Wenn Sie es nicht in Ihrer Funktion verwenden, enthält Ihre Service-Map immer noch einen Knoten für den Lambda-Service und einen für jede Lambda-Funktion. Durch Hinzufügen des SDK können Sie Ihren Funktionscode instrumentieren, um Untersegmente zu dem von Lambda aufgezeichneten Funktionssegment hinzuzufügen. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Ein Beispiel Worker für eine in Lambda instrumentierte Python-Funktion finden Sie unter.

Verwenden Sie als Nächstes das X-Ray-SDK für Python, um Downstream-Aufrufe zu instrumentieren, indem Sie die Bibliotheken Ihrer Anwendung patchen. Das SDK unterstützt die folgenden Bibliotheken.

Unterstützte Bibliotheken

- `botocore`, `boto3` — AWS SDK für Python (Boto) Instrumenten-Clients.
- `pynamodb`— Instrument PynamoDB-Version des Amazon DynamoDB-Clients.
- `aiobotocore`, `aioboto3` — Asyncio-integrierte Versionen des SDK für Python-Clients.
- `requests`, `aiohttp` — Instrumentieren Sie HTTP-Clients auf hoher Ebene.
- `httplib`, `http.client`— Instrumentieren Sie HTTP-Clients auf niedriger Ebene und die Bibliotheken auf höherer Ebene, die sie verwenden.
- `sqlite3`— SQLite Instrumenten-Clients.

- `mysql-connector-python`— Instrumentieren Sie MySQL-Clients.
- `pg8000`— Instrument Pure-Python PostgreSQL-Schnittstelle.
- `psycopg2`— Instrument PostgreSQL-Datenbankadapter.
- `pymongo`— Instrumentieren Sie MongoDB-Clients.
- `pymysql`— Instrument PyMy SQL-basierte Clients für MySQL und MariaDB.

Immer wenn Ihre Anwendung eine SQL-Datenbank oder andere HTTP-Dienste aufruft, zeichnet das SDK Informationen über den Aufruf in einem Untersegment auf. AWS AWS-Services und die Ressourcen, auf die Sie innerhalb der Dienste zugreifen, werden in der Trace-Map als Downstream-Knoten angezeigt, sodass Sie Fehler und Drosselungsprobleme bei einzelnen Verbindungen leichter identifizieren können.

Nachdem Sie das SDK verwendet haben, passen Sie sein Verhalten an, indem Sie den Rekorder und die Middleware konfigurieren. Sie können Plugins zum Festhalten von Daten über die Datenverarbeitungsressourcen, auf denen Ihre Anwendung ausgeführt wird, hinzufügen, das Samplingverhalten durch Samplingregeln anpassen und Protokollebenen einrichten, um mehr oder weniger Informationen von dem SDK in Ihren Anwendungsprotokollen zu sehen.

Zeichnen Sie zusätzliche Informationen zu Anforderungen und den Aufgaben, die Ihre Anwendung ausführt, in Anmerkungen und Metadaten auf. Anmerkungen sind einfache Schlüsselwertpaare, die für die Verwendung mit Filterausdrücken indiziert werden, damit Sie nach Ablaufverfolgen mit bestimmten Daten suchen können. Metadateneinträge sind weniger einschränkend und können ganze Objekte und Arrays aufzeichnen – alle Daten, die in eine JSON zusammengefasst werden können.

> **Anmerkungen und Metadaten**
>
> Anmerkungen und Metadaten sind beliebiger Text, den Sie Segmenten mit dem X-Ray SDK hinzufügen. Anmerkungen werden für die Verwendung mit Filterausdrücken indexiert. Metadaten werden nicht indexiert, können aber im Rohsegment mit der X-Ray-Konsole oder API angezeigt werden. Jeder, dem Sie Lesezugriff auf X-Ray gewähren, kann diese Daten einsehen.

Wenn Sie viele instrumentierten Clients in Ihrem Code haben, kann ein einzelnes Anforderungssegmente viele Untersegmente enthalten, eines für jeden Aufruf mit einem instrumentierten Client. Sie können Untersegmente organisieren und gruppieren, indem Sie Client-

Aufrufe in benutzerdefinierten Untersegmenten zusammenfassen. Sie können ein benutzerdefiniertes Untersegment für die gesamte Funktion oder einen beliebigen Bereich des Codes erstellen. Sie können dann Metadaten und Anmerkungen im Untersegment aufzeichnen, anstatt alles in das übergeordnete Segment zu schreiben.

Eine Referenzdokumentation zu den Klassen und Methoden des SDK finden Sie in der API-Referenz zum AWS X-Ray SDK for Python.

Voraussetzungen

Das X-Ray-SDK für Python unterstützt die folgenden Sprach- und Bibliotheksversionen.

- Python — 2.7, 3.4 und neuer
- Django — 1.10 und neuer
- Flask — 0.10 und neuer
- aiohttp — 2.3.0 und neuer
- AWS SDK für Python (Boto)— 1.4.0 und neuer
- botocore — 1.5.0 und neuer
- enum — 0.4.7 und neuer, für Python-Versionen 3.4.0 und älter
- jsonpickle — 1.0.0 und neuer
- setuptools — 40.6.3 und neuer
- wrapt — 1.11.0 und neuer

Abhängigkeitsmanagement

Das X-Ray-SDK für Python ist erhältlich unter`pip`.

- Package — `aws-xray-sdk`

Fügen Sie das SDK in Ihrer `requirements.txt`-Datei als abhängige Komponente hinzu.

Example requirements.txt

```
aws-xray-sdk==2.4.2
boto3==1.4.4
botocore==1.5.55
Django==1.11.3
```

Wenn Sie Elastic Beanstalk für die Bereitstellung Ihrer Anwendung verwenden, installiert Elastic Beanstalk alle Pakete automatisch. `requirements.txt`

Konfiguration des X-Ray-SDK für Python

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK für Python hat eine Klasse namens`xray_recorder`, die den globalen Rekorder bereitstellt. Sie können die globale Aufzeichnung so konfigurieren, dass die Middleware, die Segmente für eingehende HTTP-Aufrufe erstellt, angepasst wird.

Sections

- Service-Plugins
- Regeln für die Probenahme
- Protokollierung
- Konfiguration des Recorders im Code
- Konfigurieren des Recorders mit Django
- Umgebungsvariablen

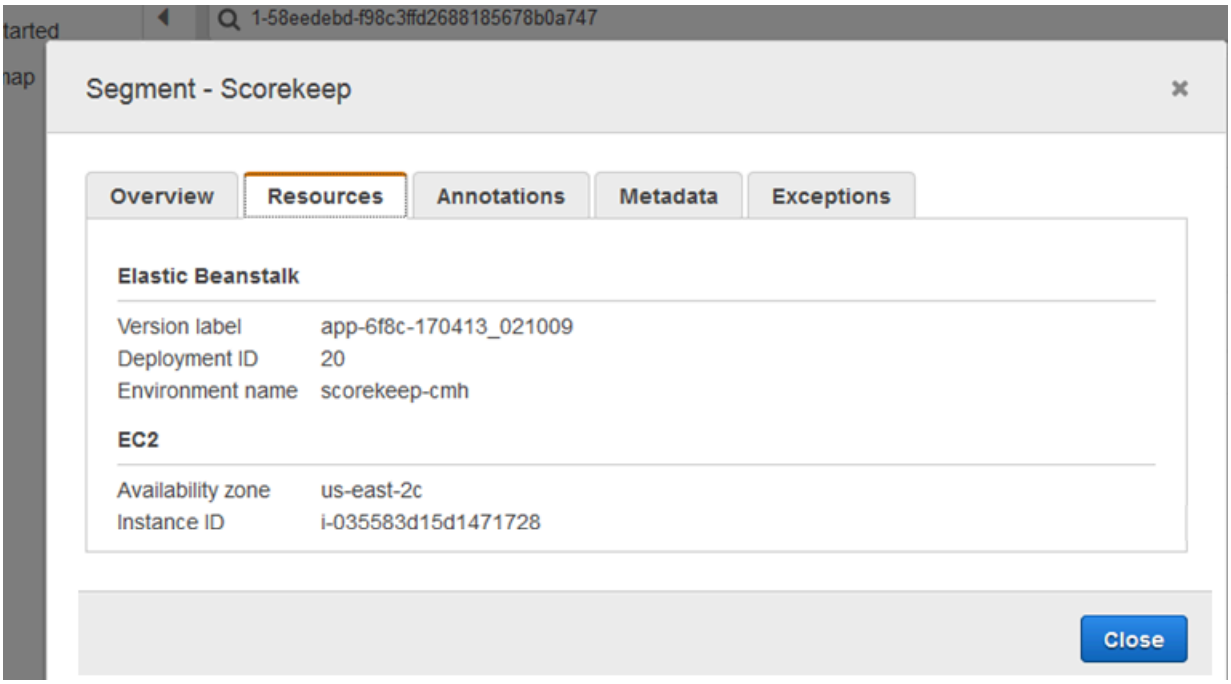
Service-Plugins

Wird verwendet`plugins`, um Informationen über den Dienst aufzuzeichnen, der Ihre Anwendung hostet.

Plugins

- Amazon EC2 — `EC2Plugin` fügt die Instance-ID, die Availability Zone und die CloudWatch Logs-Gruppe hinzu.

- Elastic Beanstalk — `ElasticBeanstalkPlugin` fügt den Umgebungsnamen, die Versionsbezeichnung und die Bereitstellungs-ID hinzu.
- Amazon ECS — `ECSPlugin` fügt die Container-ID hinzu.



Um ein Plugin zu verwenden, rufen Sie `configure` im `xray_recorder` auf.

```
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**
>
> Da `plugins` sie als Tupel übergeben werden, sollten Sie bei der Angabe eines einzelnen Plug-ins unbedingt ein `,` abschließendes Zeichen angeben. Beispiel: `plugins = ('EC2Plugin',)`

Sie können auch Umgebungsvariablen, die Vorrang über Werte im Code haben, zur Konfiguration des Recorders verwenden.

Konfigurieren Sie Plugins vor Patch-Bibliotheken, um nachgeschaltete Aufrufe aufzuzeichnen.

Das SDK verwendet auch Plugin-Einstellungen, um das `origin` Feld für das Segment festzulegen. Dies gibt den AWS Ressourcentyp an, auf dem Ihre Anwendung ausgeführt wird. Wenn Sie mehrere Plugins verwenden, verwendet das SDK die folgende Auflösungsreihenfolge, um den Ursprung zu bestimmen: ElasticBeanstalk > EKS > ECS > EC2.

Regeln für die Probenahme

Das SDK verwendet die Sampling-Regeln, die Sie in der X-Ray-Konsole definieren, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen. Die Standardregel verfolgt die erste Anfrage jede Sekunde und fünf Prozent aller weiteren Anfragen aller Dienste, die Traces an X-Ray senden. Erstellen Sie zusätzliche Regeln in der X-Ray-Konsole, um die Menge der aufgezeichneten Daten für jede Ihrer Anwendungen anzupassen.

Das SDK wendet benutzerdefinierte Regeln in der Reihenfolge an, in der sie definiert sind. Wenn eine Anfrage mehreren benutzerdefinierten Regeln entspricht, wendet das SDK nur die erste Regel an.

> **Note**
>
> Wenn das SDK X-Ray nicht erreichen kann, um Sampling-Regeln abzurufen, kehrt es zu einer lokalen Standardregel zurück, die die erste Anfrage pro Sekunde und fünf Prozent aller zusätzlichen Anfragen pro Host vorsieht. Dies kann passieren, wenn der Host nicht berechtigt ist APIs, Sampling aufzurufen, oder wenn er keine Verbindung zum X-Ray-Daemon herstellen kann, der als TCP-Proxy für API-Aufrufe durch das SDK fungiert.

Sie können das SDK auch so konfigurieren, dass Sampling-Regeln aus einem JSON-Dokument geladen werden. Das SDK kann lokale Regeln als Backup für Fälle verwenden, in denen X-Ray Sampling nicht verfügbar ist, oder ausschließlich lokale Regeln verwenden.

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
  }
}
```

Dieses Beispiel definiert eine benutzerdefinierte Regel und eine Standardregel. Die benutzerdefinierte Regel wendet eine Stichprobenrate von fünf Prozent an, ohne dass eine Mindestanzahl von Anfragen für Pfade verfolgt werden muss. `/api/move/` Die Standardregel verfolgt die erste Anfrage jede Sekunde und 10 Prozent der weiteren Anfragen.

Der Nachteil der lokalen Definition von Regeln besteht darin, dass das feste Ziel von jeder Instanz des Rekorders unabhängig angewendet wird, anstatt vom X-Ray-Dienst verwaltet zu werden. Wenn Sie mehr Hosts bereitstellen, wird die feste Rate vervielfacht, wodurch es schwieriger wird, die Menge der aufgezeichneten Daten zu kontrollieren.

Wenn aktiviert AWS Lambda, können Sie die Samplerate nicht ändern. Wenn Ihre Funktion von einem instrumentierten Dienst aufgerufen wird, werden Aufrufe, die Anfragen generierten, die von diesem Dienst abgetastet wurden, von Lambda aufgezeichnet. Wenn aktives Tracing aktiviert ist und kein Tracing-Header vorhanden ist, trifft Lambda die Stichprobenentscheidung.

Um Backup-Sampling-Regeln zu konfigurieren`xray_recorder.configure`, rufen Sie, wie im folgenden Beispiel gezeigt, auf, wobei entweder *`rules`* ein Regelwörterbuch oder der absolute Pfad zu einer JSON-Datei mit Sampling-Regeln steht.

```
xray_recorder.configure(sampling_rules=rules)
```

Um nur lokale Regeln zu verwenden, konfigurieren Sie den Recorder mit einer `LocalSampler`.

```
from aws_xray_sdk.core.sampling.local.sampler import LocalSampler
xray_recorder.configure(sampler=LocalSampler())
```

Sie können auch die globale Aufzeichnung so konfigurieren, dass das Sampling deaktiviert und alle eingehenden Anfragen instrumentiert werden.

Example main.py — Sampling deaktivieren

```
xray_recorder.configure(sampling=False)
```

Protokollierung

Das SDK verwendet das in Python integrierte `logging` Modul mit einer `WARNING` Standard-Protokollierungsebene. Erhalten Sie eine Referenz zum Logger für die `aws_xray_sdk`-Klasse und rufen Sie darauf `setLevel` auf, um die verschiedenen Protokollebenen für die Bibliothek und den Rest Ihrer Anwendung zu konfigurieren.

Example app.py — Protokollierung

```
logging.basicConfig(level='WARNING')
logging.getLogger('aws_xray_sdk').setLevel(logging.ERROR)
```

Verwenden Sie Debug-Protokolle, um Probleme wie nicht geschlossene Untersegmente zu identifizieren, wenn Sie Untersegmente manuell generieren.

Konfiguration des Recorders im Code

Zusätzliche Einstellungen finden Sie in der `configure`-Methode auf `xray_recorder`.

- `context_missing`— Auf einstellen, um `LOG_ERROR` zu verhindern, dass Ausnahmen ausgelöst werden, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, obwohl kein Segment geöffnet ist.
- `daemon_address`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest.
- `service`— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet.
- `plugins`— Notieren Sie Informationen über die AWS Ressourcen Ihrer Anwendung.
- `sampling`— Auf einstellen, `False` um die Probenahme zu deaktivieren.
- `sampling_rules`— Legen Sie den Pfad der JSON-Datei fest, die Ihre Sampling-Regeln enthält.

Example main.py — Deaktiviert Ausnahmen, die im Kontext fehlen

```
from aws_xray_sdk.core import xray_recorder

xray_recorder.configure(context_missing='LOG_ERROR')
```

Konfigurieren des Recorders mit Django

Wenn Sie das Django-Framework verwenden, können Sie die `settings.py`-Datei von Django zum Konfigurieren von Optionen in der globalen Aufzeichnung verwenden.

- `AUTO_INSTRUMENT`(Nur Django) — Zeichnet Untersegmente für integrierte Datenbank- und Vorlagen-Rendering-Operationen auf.
- `AWS_XRAY_CONTEXT_MISSING`— Auf einstellen, um `LOG_ERROR` zu vermeiden, dass Ausnahmen ausgelöst werden, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, wenn kein Segment geöffnet ist.
- `AWS_XRAY_DAEMON_ADDRESS`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest.
- `AWS_XRAY_TRACING_NAME`— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet.
- `PLUGINS`— Notieren Sie Informationen über die AWS Ressourcen Ihrer Anwendung.
- `SAMPLING`— Auf einstellen, `False` um die Probenahme zu deaktivieren.
- `SAMPLING_RULES`— Legen Sie den Pfad der JSON-Datei fest, die Ihre Sampling-Regeln enthält.

Um die Konfiguration des Recorders in `settings.py` zu aktivieren, fügen Sie die Django-Middleware zur Liste der installierten Apps hinzu.

Example settings.py — Installierte Apps

```
INSTALLED_APPS = [
    ...
    'django.contrib.sessions',
    'aws_xray_sdk.ext.django',
]
```

Konfigurieren Sie die verfügbaren Einstellungen in einem dict-Objekt mit dem Namen `XRAY_RECORDER`.

Example settings.py — Installierte Apps

```
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,
}
```

Umgebungsvariablen

Sie können Umgebungsvariablen verwenden, um das X-Ray SDK für Python zu konfigurieren. Das SDK unterstützt die folgenden Variablen:

- `AWS_XRAY_TRACING_NAME`— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet. Überschreibt den Servicenamen, den Sie programmgesteuert festgelegt haben.
- `AWS_XRAY_SDK_ENABLED`— Wenn auf gesetzt`false`, wird das SDK deaktiviert. Das SDK ist standardmäßig aktiviert, es sei denn, die Umgebungsvariable ist auf „false“ festgelegt.
  - Wenn diese Option deaktiviert ist, generiert die globale Aufzeichnung automatisch Dummy-Segmente und Untersegmente, die nicht an den Daemon gesendet werden, und das automatische Patchen wird deaktiviert. Middlewares werden als Wrapper über der globalen Aufzeichnung geschrieben. Auch alle Generierungen von Segmenten und Untersegmenten über die Middleware werden zu Dummy-Segmenten und Dummy-Untersegmenten.
  - Legen Sie den Wert `AWS_XRAY_SDK_ENABLED` über die Umgebungsvariable oder durch direkte Interaktion mit dem Objekt `global_sdk_config` aus der `aws_xray_sdk`-Bibliothek fest. Einstellungen der Umgebungsvariablen überschreiben diese Interaktionen.
- `AWS_XRAY_DAEMON_ADDRESS`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest. Standardmäßig verwendet `127.0.0.1:2000` das SDK sowohl Trace-Daten (UDP) als auch Sampling-Daten (TCP). Verwenden Sie diese Variable, wenn Sie den Daemon so konfiguriert haben, dass er auf einem anderen Port lauscht oder wenn er auf einem anderen Host läuft.

  Format

  - Derselbe Port — *`address`*`:`*`port`*
  - Verschiedene Anschlüsse — `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_CONTEXT_MISSING`— Auf einstellen, `RUNTIME_ERROR` um Ausnahmen auszulösen, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, obwohl kein Segment geöffnet ist.

  Zulässige Werte

  - `RUNTIME_ERROR`— Löst eine Laufzeitausnahme aus.
  - `LOG_ERROR`— Fehler protokollieren und fortfahren (Standard).
  - `IGNORE_ERROR`— Fehler ignorieren und fortfahren.

Fehler im Zusammenhang mit fehlenden Segmenten oder Untersegmenten können auftreten, wenn Sie versuchen, einen instrumentierten Client in Startcode zu verwenden, der ausgeführt wird, wenn keine Anfrage geöffnet ist, oder in Code, der einen neuen Thread erzeugt.

Umgebungsvariablen überschreiben Werte im Code.

Nachverfolgung eingehender Anfragen mit dem X-Ray SDK für Python-Middleware

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Sie die Middleware zu Ihrer Anwendung hinzufügen und einen Segmentnamen konfigurieren, erstellt das X-Ray SDK für Python für jede gesampelte Anfrage ein Segment. Dieses Segment umfasst Dauer, Methode und Status der HTTP-Anforderung. Die zusätzliche Instrumentierung schafft Untersegmente zu diesem Segment.

Das X-Ray SDK für Python unterstützt die folgende Middleware zur Instrumentierung eingehender HTTP-Anfragen:

- Django
- Flask
- Bottle

> **Note**
>
> Für AWS Lambda Funktionen erstellt Lambda für jede abgetastete Anfrage ein Segment. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Ein Beispiel Worker für eine in Lambda instrumentierte Python-Funktion finden Sie unter.

Für Skripts oder Python-Anwendungen auf anderen Frameworks können Sie Segmente manuell erstellen.

Jedes Segment hat einen Namen, der Ihre Anwendung in der Service Map identifiziert. Das Segment kann statisch benannt werden, oder Sie können das SDK so konfigurieren, dass es dynamisch auf der Grundlage des Host-Headers in der eingehenden Anfrage benannt wird. Mit der dynamischen Benennung können Sie Traces auf der Grundlage des Domainnamens in der Anfrage gruppieren und einen Standardnamen anwenden, wenn der Name nicht einem erwarteten Muster entspricht (z. B. wenn der Host-Header gefälscht ist).

> **Weitergeleitete Anfragen**
>
> Wenn ein Load Balancer oder ein anderer Vermittler eine Anfrage an Ihre Anwendung weiterleitet, nimmt X-Ray die Client-IP aus dem `X-Forwarded-For` Header in der Anfrage und nicht aus der Quell-IP im IP-Paket. Die Client-IP, die für eine weitergeleitete Anfrage aufgezeichnet wird, kann gefälscht sein und sollte daher nicht als vertrauenswürdig eingestuft werden.

Wenn eine Anfrage weitergeleitet wird, legt das SDK ein zusätzliches Feld im Segment fest, um dies anzuzeigen. Wenn das Segment das Feld enthält, das auf `x_forwarded_for` gesetzt ist`true`, wurde die Client-IP aus dem `X-Forwarded-For` Header in der HTTP-Anfrage übernommen.

Die Middleware erzeugt für jede eingehende Anfrage ein Segment mit einem `http`-Block mit folgenden Informationen:

- HTTP-Methode — GET, POST, PUT, DELETE usw.
- Client-Adresse — Die IP-Adresse des Clients, der die Anfrage gesendet hat.
- Antwortcode — Der HTTP-Antwortcode für die abgeschlossene Anfrage.

- Timing — Die Startzeit (als die Anfrage empfangen wurde) und die Endzeit (als die Antwort gesendet wurde).
- Benutzeragent — Der `user-agent` aus der Anfrage.
- Länge des Inhalts — Die Länge `content-length` der Antwort.

Sections

Hinzufügen der Middleware zu Ihrer Anwendung (Django)

Fügen Sie die Middleware zur `MIDDLEWARE`-Liste in Ihrer `settings.py`-Datei hinzu. Die X-Ray-Middleware sollte die erste Zeile in Ihrer `settings.py`-Datei darstellen, um sicherzugehen, dass Anfragen, die in anderen Middlewares fehlschlagen, aufgezeichnet werden.

Example settings.py — X-Ray-SDK für Python-Middleware

```
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'
]
```

Fügen Sie die X-Ray SDK Django-App zur `INSTALLED_APPS` Liste in Ihrer `settings.py` Datei hinzu. Dadurch kann der Röntgenrekorder während des Starts Ihrer App konfiguriert werden.

Example settings.py — X-Ray-SDK für die Python-Django-App

```
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',
]
```

Konfigurieren Sie einen Segmentnamen in Ihrer `settings.py`-Datei.

Example settings.py — Segmentname

```
XRAY_RECORDER = {
    'AWS_XRAY_TRACING_NAME': 'My application',
    'PLUGINS': ('EC2Plugin',),
}
```

Dadurch wird der X-Recorder angewiesen, Anfragen, die von Ihrer Django-Anwendung bedient werden, mit der Standard-Abtastrate zu verfolgen. Sie können den Recorder in Ihrer Django-Einstellungsdatei konfigurieren, um benutzerdefinierte Samplingregeln anzuwenden oder andere Einstellungen zu ändern.

> **Note**
>
> Da `plugins` sie als Tupel übergeben werden, achten Sie darauf, dass Sie `,` bei der Angabe eines einzelnen Plugins ein abschließendes Zeichen angeben. Beispiel: `plugins = ('EC2Plugin',)`

Hinzufügen der Middleware zu Ihrer Anwendung (Flask)

Um Ihre Flask-Anwendung zu instrumentieren, konfigurieren Sie zunächst einen Segmentnamen im `xray_recorder`. Verwenden Sie dann die `XRayMiddleware`-Funktion zum Patchen Ihrer Flask-Anwendung im Code.

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)
```

Dadurch wird der Röntgenrekorder angewiesen, Anfragen, die von Ihrer Flask-Anwendung bedient werden, mit der Standard-Abtastrate zu verfolgen. Sie können den Recorder im Code konfigurieren, um benutzerdefinierte Samplingregeln anzuwenden oder andere Einstellungen zu ändern.

Hinzufügen der Middleware zu Ihrer Anwendung (Bottle)

Um Ihre Bottle-Anwendung zu instrumentieren, konfigurieren Sie zunächst einen Segmentnamen im `xray_recorder`. Verwenden Sie dann die `XRayMiddleware`-Funktion zum Patchen Ihrer Bottle-Anwendung im Code.

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))
```

Dadurch wird der Röntgenrekorder angewiesen, Anfragen, die von Ihrer Bottle-Anwendung gesendet wurden, mit der Standard-Abtastrate zu verfolgen. Sie können den Recorder im Code konfigurieren, um benutzerdefinierte Samplingregeln anzuwenden oder andere Einstellungen zu ändern.

Manuelle Instrumentierung von Python-Code

Wenn Sie weder Django noch Flask verwenden, können Sie Segmente manuell erstellen. Sie können für jede eingehende Anfrage ein Segment erstellen oder Segmente rund um gepatchte HTTP- oder AWS SDK-Clients erstellen, um dem Rekorder Kontext für das Hinzufügen von Untersegmenten zu bieten.

Example main.py — Manuelle Instrumentierung

```
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()
```

Konfiguration einer Segmentbenennungsstrategie

AWS X-Ray verwendet einen Dienstnamen, um Ihre Anwendung zu identifizieren und sie von den anderen Anwendungen, Datenbanken, externen AWS Ressourcen und Ressourcen zu unterscheiden APIs, die Ihre Anwendung verwendet. Wenn das X-Ray SDK Segmente für eingehende Anfragen generiert, zeichnet es den Dienstnamen Ihrer Anwendung im Namensfeld des Segments auf.

Das X-Ray SDK kann Segmente nach dem Hostnamen im HTTP-Anforderungsheader benennen. Dieser Header kann jedoch gefälscht sein, was zu unerwarteten Knoten in Ihrer Service Map führen kann. Um zu verhindern, dass das SDK Segmente aufgrund von Anfragen mit gefälschten Host-Headern falsch benennt, müssen Sie einen Standardnamen für eingehende Anfragen angeben.

Wenn Ihre Anwendung Anfragen für mehrere Domänen bearbeitet, können Sie das SDK so konfigurieren, dass es eine dynamische Benennungsstrategie verwendet, um dies in Segmentnamen widerzuspiegeln. Eine dynamische Benennungsstrategie ermöglicht es dem SDK, den Hostnamen für Anfragen zu verwenden, die einem erwarteten Muster entsprechen, und den Standardnamen auf Anfragen anzuwenden, bei denen dies nicht der Fall ist.

Beispielsweise können Sie über eine einzige Anwendung verfügen, die Anfragen an drei Subdomänen — `www.example.com`api.example.com, und — bedient. `static.example.com` Sie können eine dynamische Benennungsstrategie mit dem Muster verwenden`*.example.com`, um Segmente für jede Subdomain mit einem anderen Namen zu identifizieren, was zu drei Dienstknoten auf der Service-Map führt. Wenn Ihre Anwendung Anfragen mit einem Hostnamen empfängt, der nicht dem Muster entspricht, wird auf der Service Map ein vierter Knoten mit einem von Ihnen angegebenen Fallback-Namen angezeigt.

Wenn Sie denselben Namen für alle Anfragesegmente verwenden möchten, geben Sie bei der Konfiguration des Recorders den Namen Ihrer Anwendung, wie in den vorherigen Abschnitten gezeigt, ein.

Eine dynamische Benennungsstrategie definiert ein Muster, dem Hostnamen entsprechen sollten, sowie einen Standardnamen, der verwendet wird, wenn der Hostname in der HTTP-Anforderung nicht mit diesem Muster übereinstimmt. Zur dynamischen Segmentbenennung fügen Sie die `DYNAMIC_NAMING`-Einstellung zu Ihrer settings.py-Datei hinzu.

Example settings.py — Dynamische Benennung

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_TRACING_NAME': 'My application',
    'DYNAMIC_NAMING': '*.example.com',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin')
}
```

Sie können "*" im Muster verwenden, um eine Übereinstimmung mit einer beliebigen Zeichenfolge zu erzielen, oder "?" für die Übereinstimmung mit einem einzelnen Zeichen. Für Flask konfigurieren Sie den Recorder im Code.

Example main.py — Segmentname

```
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='My application')
xray_recorder.configure(dynamic_naming='*.example.com')
```

> **Note**
>
> Sie können den mit der `AWS_XRAY_TRACING_NAME`-Umgebungsvariablen in Code definierten standardmäßigen Dienstnamen überschreiben.

Patchen von Bibliotheken zum Instrumentieren von nachgelagerten Aufrufen

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon

Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Um Downstream-Aufrufe zu instrumentieren, verwenden Sie das X-Ray-SDK für Python, um die Bibliotheken zu patchen, die Ihre Anwendung verwendet. Das X-Ray-SDK für Python kann die folgenden Bibliotheken patchen.

Unterstützte Bibliotheken

- `botocore`, `boto3` — AWS SDK für Python (Boto) Instrumenten-Clients.
- `pynamodb`— Instrumentieren Sie die Version des Amazon DynamoDB-Clients von PynamoDB.
- `aiobotocore`, `aioboto3` — Instrument `asyncio` -integrierte Versionen des SDK für Python-Clients.
- `requests`, `aiohttp` — Instrumentieren Sie HTTP-Clients auf hoher Ebene.
- `httplib`, `http.client`— Instrumentieren Sie HTTP-Clients auf niedriger Ebene und die Bibliotheken auf höherer Ebene, die sie verwenden.
- `sqlite3`— SQLite Instrumenten-Clients.
- `mysql-connector-python`— Instrumentieren Sie MySQL-Clients.
- `pg8000`— Instrument Pure-Python PostgreSQL-Schnittstelle.
- `psycopg2`— Instrument PostgreSQL-Datenbankadapter.
- `pymongo`— Instrumentieren Sie MongoDB-Clients.
- `pymysql`— Instrument PyMy SQL-basierte Clients für MySQL und MariaDB.

Wenn Sie eine gepatchte Bibliothek verwenden, erstellt das X-Ray SDK für Python ein Untersegment für den Aufruf und zeichnet Informationen aus der Anfrage und Antwort auf. Für das SDK muss ein Segment zur Verfügung stehen, damit es ein Untersegment aus der SDK-Middleware oder aus AWS Lambda erstellen kann.

Note

Wenn Sie SQLAlchemy ORM verwenden, können Sie Ihre SQL-Abfragen instrumentieren, indem Sie die SDK-Version der SQLAlchemy Sitzungs- und Abfrageklassen importieren. Anweisungen finden Sie unter SQLAlchemy ORM verwenden.

Um alle verfügbaren Bibliotheken zu patchen, verwenden Sie die `patch_all`-Funktion in `aws_xray_sdk.core`. Einige Bibliotheken, wie z. B. `httplib` und `urllib`, müssen möglicherweise doppelte Patches durch Aufruf von `patch_all(double_patch=True)` aktivieren.

Example main.py — Patchen Sie alle unterstützten Bibliotheken

```
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()
```

Rufen Sie zum Patchen einer einzelnen Bibliothek `patch` mit einem Tupel des Bibliotheksnamens auf. Um dies zu erreichen, müssen Sie eine einzelne Elementliste bereitstellen.

Example main.py — Patchen Sie bestimmte Bibliotheken

```
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**
>
> In einigen Fällen stimmt der Schlüssel, mit dem Sie eine Bibliothek patchen, nicht mit dem Namen der Bibliothek überein. Einige Schlüssel dienen als Aliase für eine oder mehrere Bibliotheken.
>
> Aliase für Bibliotheken
>
> - `httplib` – `httplib` und `http.client`

- `mysql` – `mysql-connector-python`

Ablaufverfolgungskontext für asynchrone Vorgänge

Für `asyncio` integrierte Bibliotheken oder um Untersegmente für asynchrone Funktionen zu erstellen, müssen Sie auch das X-Ray SDK für Python mit einem asynchronen Kontext konfigurieren. Importieren Sie die `AsyncContext` Klasse und übergeben Sie eine Instanz davon an den X-Ray-Recorder.

> **Note**
>
> Bibliotheken für die Unterstützung von Web-Frameworks, wie beispielsweise AIOHTTP, werden nicht über das Modul `aws_xray_sdk.core.patcher` bearbeitet. Sie werden nicht im `patcher`-Katalog unterstützter Bibliotheken angezeigt.

Example main.py — Patch 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)
```

Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK für Python

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon

> Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Aufrufe tätigt, um Daten AWS-Services zu speichern, in eine Warteschlange zu schreiben oder Benachrichtigungen zu senden, verfolgt das X-Ray SDK für Python die Aufrufe im Downstream in Untersegmenten. Verfolgte Ressourcen AWS-Services und Ressourcen, auf die Sie innerhalb dieser Services zugreifen (z. B. ein Amazon S3 S3-Bucket oder eine Amazon SQS SQS-Warteschlange), werden als Downstream-Knoten auf der Trace-Map in der X-Ray-Konsole angezeigt.

Das X-Ray-SDK für Python instrumentiert automatisch alle AWS SDK-Clients, wenn Sie die `botocore` Bibliothek patchen. Einzelne Clients können nicht instrumentiert werden.

Für alle Dienste können Sie den Namen der aufgerufenen API in der X-Ray-Konsole sehen. Für eine Untergruppe von Diensten fügt das X-Ray SDK dem Segment Informationen hinzu, um die Service Map detaillierter zu gestalten.

Wenn Sie beispielsweise einen Aufruf mit einem instrumentierten DynamoDB-Client tätigen, fügt das SDK den Tabellennamen dem Segment für Aufrufe hinzu, die auf eine Tabelle abzielen. In der Konsole wird jede Tabelle als separater Knoten in der Service Map angezeigt, mit einem generischen DynamoDB-Knoten für Aufrufe, die nicht auf eine Tabelle abzielen.

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements

```
{
  "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",
  }
}
```

Wenn Sie auf benannte Ressourcen zugreifen, werden durch Aufrufe der folgenden Services weitere Knoten in der Service-Übersicht erstellt. Durch Aufrufe, die keinen bestimmten Ressourcen gelten, wird ein generischer Knoten für den Service erstellt.

- Amazon DynamoDB — Tabellenname
- Amazon Simple Storage Service — Bucket und Schlüsselname
- Amazon Simple Queue Service — Name der Warteschlange

Verfolgen von Aufrufen an Downstream-HTTP-Webservices mithilfe des X-Ray SDK für Python

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Microservices oder öffentliches HTTP aufruft APIs, können Sie das X-Ray SDK für Python verwenden, um diese Aufrufe zu instrumentieren und die API als Downstream-Service zum Service Graph hinzuzufügen.

Um HTTP-Clients zu instrumentieren, patchen Sie die Bibliothek, mit der Sie ausgehende Aufrufe durchführen. Wenn Sie `requests` oder den integrierten HTTP-Client von Python verwenden, brauchen Sie nichts weiter zu tun. Bei `aiohttp` müssen Sie zudem den Recorder mit einem asynchronen Kontext konfigurieren.

Wenn Sie die `aiohttp` 3-Client-API verwenden, müssen Sie auch die `ClientSession` mit einer Instance der Nachverfolgungskonfiguration konfigurieren, die vom SDK bereitgestellt wird.

Example **`aiohttp`** 3 Client-API

```
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()
```

Wenn Sie einen Aufruf einer Downstream-Web-API instrumentieren, zeichnet das X-Ray-SDK für Python ein Untersegment auf, das Informationen über die HTTP-Anfrage und -Antwort enthält. X-Ray verwendet das Untersegment, um ein abgeleitetes Segment für die Remote-API zu generieren.

Example Untersegment für einen nachgelagerten HTTP-Aufruf

```
{
  "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 Abgeleitetes Segment für einen nachgelagerten HTTP-Anruf

```
{
  "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
}
```

Generieren von benutzerdefinierten Untersegmenten mit dem X-Ray SDK für Python

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Untersegmente erweitern das Segment eines Traces um Details über die Arbeit, die zur Bearbeitung einer Anfrage geleistet wurde. Jedes Mal, wenn Sie einen Anruf mit einem instrumentierten Client tätigen, zeichnet das X-Ray-SDK die in einem Untersegment generierten Informationen auf. Sie können zusätzliche Untersegmente erstellen, um andere Untersegmente zu gruppieren, die Leistung eines Codeabschnitts zu messen oder Anmerkungen und Metadaten aufzuzeichnen.

Um Untersegmente zu verwalten, verwenden Sie die Methoden `begin_subsegment` und `end_subsegment`.

Example main.py — Benutzerdefiniertes Untersegment

```
from aws_xray_sdk.core import xray_recorder
```

```
subsegment = xray_recorder.begin_subsegment('annotations')
subsegment.put_annotation('id', 12345)
xray_recorder.end_subsegment()
```

Um ein Untersegment für eine synchrone Funktion zu erstellen, verwenden Sie den `@xray_recorder.capture`-Decorator. Sie können einen Namen für das Untersegment an die Erfassungsfunktion übergeben oder diesen weglassen und den Funktionsnamen verwenden.

Example main.py — Funktionsuntersegment

```
from aws_xray_sdk.core import xray_recorder

@xray_recorder.capture('## create_user')
def create_user():
...
```

Verwenden Sie bei einer asynchronen Funktion den `@xray_recorder.capture_async`-Decorator und übergeben Sie einen asynchronen Kontext an den Recorder.

Example main.py — Untersegment einer asynchronen Funktion

```
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()
```

Wenn Sie ein Untersegment innerhalb eines Segments oder eines anderen Untersegments erstellen, generiert das X-Ray SDK für Python eine ID dafür und zeichnet die Start- und Endzeit auf.

Example Untersegment mit Metadaten

```
"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"
    }
  },
```

Hinzufügen von Anmerkungen und Metadaten zu Segmenten mit dem X-Ray SDK für Python

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können zusätzliche Informationen über Anfragen, die Umgebung oder Ihre Anwendung mit Anmerkungen und Metadaten aufzeichnen. Sie können Anmerkungen und Metadaten zu den Segmenten hinzufügen, die das X-Ray SDK erstellt, oder zu benutzerdefinierten Untersegmenten, die Sie erstellen.

Anmerkungen sind Schlüssel-Wert-Paare mit Zeichenfolgen-, Zahlen- oder booleschen Werten. Anmerkungen sind für die Verwendung mit Filterausdrücken indexiert. Berücksichtigen Sie Anmerkungen, um Daten zur Gruppierung von Ablaufverfolgungen in der Konsole zu verwenden, oder wenn Sie die `GetTraceSummaries`-API aufrufen.

Metadaten sind Schlüssel-Wert-Paare, die Werte beliebigen Typs enthalten können, einschließlich Objekte und Listen, aber nicht für die Verwendung mit Filterausdrücken indexiert sind. Verwenden Sie Metadaten, um zusätzliche Daten aufzuzeichnen, die Sie im Trace speichern möchten, aber nicht für die Suche verwenden müssen.

Zusätzlich zu Anmerkungen und Metadaten können Sie auch Benutzer-ID-Zeichenfolgen in Segmenten aufzeichnen. Benutzer IDs werden in einem separaten Feld in Segmenten aufgezeichnet und für die Verwendung bei der Suche indexiert.

Sections

Anmerkungen mit dem X-Ray SDK für Python aufnehmen

Verwenden Sie Anmerkungen, um Informationen zu Segmenten oder Untersegmenten, die zur Suche indiziert werden sollten, aufzuzeichnen.

Anmerkung zu Anforderungen

- Schlüssel — Der Schlüssel für eine X-Ray-Anmerkung kann bis zu 500 alphanumerische Zeichen enthalten. Sie können keine anderen Leerzeichen oder Symbole als einen Punkt oder Punkt (.) verwenden
- Werte — Der Wert für eine X-Ray-Anmerkung kann bis zu 1.000 Unicode-Zeichen enthalten.
- Die Anzahl der Anmerkungen — Sie können bis zu 50 Anmerkungen pro Spur verwenden.

So zeichnen Sie Anmerkungen auf

1. Eine Referenz des aktuellen Segments oder Untersegments finden Sie unter `xray_recorder`.

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_segment()
   ```

   oder

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_subsegment()
   ```

2. Rufen Sie `put_annotation` mit einem Aktivierungsschlüssel und einem booleschen Wert oder einem Zeichenfolgenwert auf.

   ```
   document.put_annotation("mykey", "my value");
   ```

Das folgende Beispiel zeigt, wie Sie `putAnnotation` mit einem String-Schlüssel aufrufen, der einen Punkt und einen booleschen Wert, eine Zahl oder einen String-Wert enthält.

```
document.putAnnotation("testkey.test", "my value");
```

Alternativ können Sie auch die `put_annotation`-Methode im `xray_recorder` verwenden. Mit dieser Methode werden Anmerkungen zum aktuellen Untersegment oder zum Segment aufgezeichnet, wenn kein Untersegment geöffnet ist.

```
xray_recorder.put_annotation("mykey", "my value");
```

Das SDK zeichnet Anmerkungen als Schlüssel-Wert-Paare in einem `annotations`-Objekt im Segmentdokument auf. Wenn `put_annotation` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Nutzen Sie das `annotation[`*`key`*`]`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen durch Anmerkungen mit bestimmten Werten zu finden.

Metadaten mit dem X-Ray SDK für Python aufzeichnen

> **Warning**
>
> Fügen Sie dem X-Ray SDK für Python keine Objekte mit Zirkelverweisen als Metadatenwerte hinzu. Diese Objekte können nicht in JSON serialisiert werden und können Endlosschleifen im SDK erzeugen. Vermeiden Sie außerdem, große, komplexe Objekte als Metadaten hinzuzufügen, um Leistungsprobleme zu vermeiden.

Verwenden Sie Metadaten, um Segment- oder Untersegmentinformationen aufzuzeichnen, die nicht zur Suche indiziert werden müssen. Metadatenwerte sind Zeichenfolgen, Zahlen, boolesche Werte oder andere Objekte, die in Form eines JSON-Objekts oder eines Arrays angeordnet sein können.

So zeichnen Sie Metadaten auf

1. Eine Referenz des aktuellen Segments oder Untersegments finden Sie unter `xray_recorder`.

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   ```

```
document = xray_recorder.current_segment()
```

oder

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. Rufen Sie `put_metadata` mit einem Zeichenfolgenschlüssel, einem booleschen Wert, einer Zahl, einer Zeichenfolge oder einem Objektwert und einem Zeichenfolgen-Namespace auf.

```
document.put_metadata("my key", "my value", "my namespace");
```

oder

Rufen Sie `put_metadata` nur mit einem Aktivierungsschlüssel und einem Wert auf.

```
document.put_metadata("my key", "my value");
```

Alternativ können Sie auch die `put_metadata`-Methode im `xray_recorder` verwenden. Mit dieser Methode werden Metadaten zum aktuellen Untersegment oder zum Segment aufgezeichnet, wenn kein Untersegment geöffnet ist.

```
xray_recorder.put_metadata("my key", "my value");
```

Wenn Sie keinen Namespace angeben, verwendet SDK `default`. Wenn `put_metadata` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Benutzer IDs mit dem X-Ray SDK für Python aufzeichnen

Zeichnen Sie Segmente von Benutzern IDs auf Anfrage auf, um den Benutzer zu identifizieren, der die Anfrage gesendet hat.

Um den Benutzer aufzuzeichnen IDs

1. Eine Referenz des aktuellen Segments finden Sie unter `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
```

```
...
document = xray_recorder.current_segment()
```

2. Rufen Sie `setUser` mit einer Zeichenfolgen-ID des Benutzers auf, der die Anforderung gesendet hat.

```
document.set_user("U12345");
```

Sie können `set_user` in Ihrem Controller aufrufen, um die Benutzer-ID aufzuzeichnen, sobald die Anwendung mit der Bearbeitung einer Anfrage beginnt.

Nutzen Sie das `user`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen einer Benutzer-ID zu finden.

Instrumentieren von in serverlosen Umgebungen bereitgestellten Web-Frameworks

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das AWS X-Ray SDK für Python unterstützt die Instrumentierung von Web-Frameworks, die in serverlosen Anwendungen eingesetzt werden. Die Cloud ist von Natur aus serverlos und dies bietet Ihnen die Möglichkeit, immer mehr Ihrer betrieblichen Verantwortung auf AWS zu verlagern. Der unschätzbare Nutzen für Sie sind mehr Agilität und Innovation.

Eine serverlose Architektur ist ein Softwareanwendungsmodell, mit dem Sie Anwendungen und Services entwickeln und ausführen können, ohne über Server nachdenken zu müssen. Für Sie verringern sich die Aufgaben der Infrastrukturverwaltung wie die Bereitstellung von Servern oder Clustern, Patching, Betriebssystemwartung und Kapazitätsbereitstellung. Sie können serverlose Lösungen für praktisch jeden Anwendungstyp oder Backend-Service erstellen und alles, was zum

Ausführen und Skalieren Ihrer Anwendung mit hoher Verfügbarkeit erforderlich ist, wird für Sie durchgeführt.

Dieses Tutorial zeigt Ihnen, wie Sie ein Web-Framework wie Flask oder Django, das in einer serverlosen Umgebung bereitgestellt wird, automatisch AWS X-Ray instrumentieren. Die X-Ray-Instrumentierung der Anwendung ermöglicht es Ihnen, alle getätigten Downstream-Aufrufe, angefangen von Amazon API Gateway über Ihre AWS Lambda Funktion, bis hin zu den ausgehenden Aufrufen Ihrer Anwendung zu sehen.

Das X-Ray-SDK für Python unterstützt die folgenden Python-Anwendungsframeworks:

- Flask-Version 0.8 oder höher
- Django-Version 1.0 oder höher

In diesem Tutorial wird eine serverlose Beispielanwendung entwickelt, die auf Lambda bereitgestellt und von API Gateway aufgerufen wird. In diesem Tutorial wird Zappa verwendet, um die Anwendung automatisch auf Lambda bereitzustellen und den API-Gateway-Endpunkt zu konfigurieren.

Voraussetzungen

- Zappa
- Python — Version 2.7 oder 3.6.
- AWS CLI— Stellen Sie sicher, dass Ihr Konto mit dem Konto konfiguriert AWS CLI ist und AWS-Region in dem Sie Ihre Anwendung bereitstellen werden.
- Pip
- Virtualenv

Schritt 1: Erstellen einer Umgebung

In diesem Schritt erstellen Sie eine virtuelle Umgebung mit `virtualenv` zum Hosten einer Anwendung.

1. Erstellen Sie mit dem AWS CLI ein Verzeichnis für die Anwendung. Wechseln Sie anschließend zum neuen Verzeichnis.

```
mkdir serverless_application
cd serverless_application
```

2. Erstellen Sie dann eine virtuelle Umgebung in Ihrem neuen Verzeichnis. Aktivieren Sie sie mit dem folgenden Befehl.

```
# Create our virtual environment
virtualenv serverless_env

# Activate it
source serverless_env/bin/activate
```

3. Installieren Sie X-Ray, Flask, Zappa und die Requests-Bibliothek in Ihrer Umgebung.

```
# Install X-Ray, Flask, Zappa, and Requests into your environment
pip install aws-xray-sdk flask zappa requests
```

4. Fügen Sie Anwendungscode zum Verzeichnis `serverless_application` hinzu. In diesem Beispiel bauen wir auf dem Flask-Beispiel Hallo Welt auf.

   Erstellen Sie in dem Verzeichnis `serverless_application` eine Datei namens `my_app.py`. Fügen Sie anschließend mit einem Texteditor die folgenden Befehle hinzu. Diese Anwendung instrumentiert die Anfragebibliothek, führt Patches für die Middleware der Flask-Anwendung aus und öffnet den Endpunkt `'/'`.

```
# 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
```

Schritt 2: Erstellen und Bereitstellen einer Zappa-Umgebung

In diesem Schritt verwenden Sie Zappa, um automatisch einen API-Gateway-Endpunkt zu konfigurieren und ihn dann auf Lambda bereitzustellen.

1. Initialisieren Sie Zappa über das Verzeichnis `serverless_application`. In diesem Beispiel verwenden wir die Standardeinstellungen. Wenn Sie jedoch über individuelle Voreinstellungen verfügen, zeigt Zappa Anleitungen für die Konfiguration an.

   ```
   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. X-Ray aktivieren. Öffnen Sie die Datei `zappa_settings.json` und stellen Sie sicher, dass sie dem Beispiel ähnelt.

   ```
   {
       "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. Fügen Sie `"xray_tracing": true` als Eintrag zur Konfigurationsdatei hinzu.

```
{
    "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. Stellen Sie die Anwendung bereit. Dadurch wird der API-Gateway-Endpunkt automatisch konfiguriert und Ihr Code auf Lambda hochgeladen.

```
zappa deploy
```

```
...
Deploying API Gateway..
Deployment complete!: https://**********.execute-api.us-west-2.amazonaws.com/dev
```

Schritt 3: X-Ray Tracing für API Gateway aktivieren

In diesem Schritt interagieren Sie mit der API Gateway Gateway-Konsole, um das X-Ray-Tracing zu aktivieren.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die API Gateway Gateway-Konsole unter https://console.aws.amazon.com/apigateway/.
2. Suchen Sie Ihre neu generierte API. Sie sollte in etwa so aussehen: `serverless-exam-dev`.
3. Wählen Sie Stages.
4. Wählen Sie den Namen der Bereitstellungsstufe. Der Standardwert ist `dev`.
5. Wählen Sie auf der Registerkarte Logs/Tracing (Protokolle/Nachverfolgung) die Option Enable X-Ray Tracing (X-Ray-Tracing aktivieren) aus.
6. Wählen Sie Save Changes.

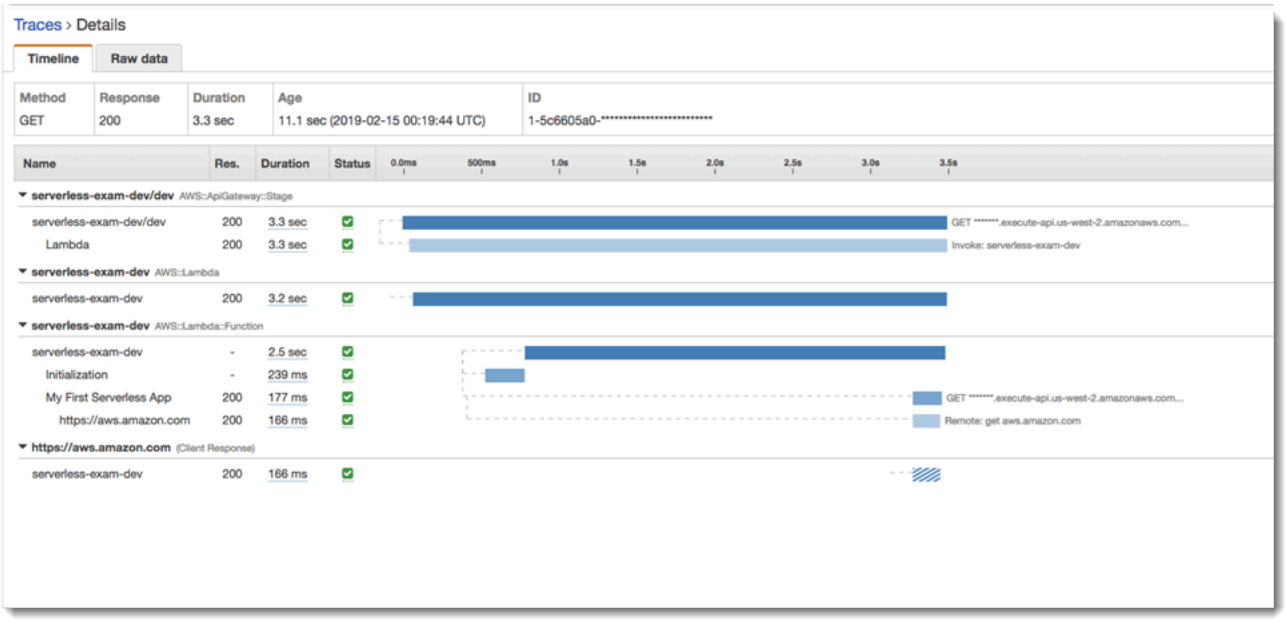
7. Rufen Sie den Endpunkt in Ihrem Browser auf. Wenn Sie die Beispielanwendung `Hello World` verwendet haben, sollte Folgendes angezeigt werden.

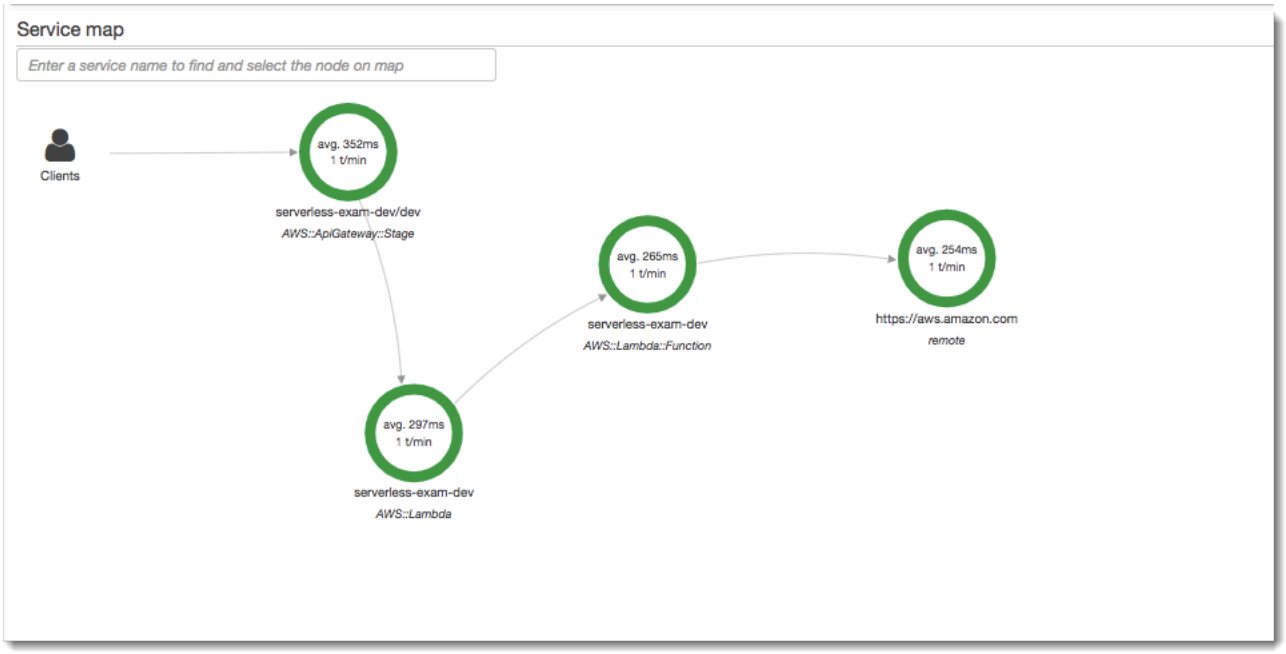
```
"Hello, World: https://aws.amazon.com/"
```

Schritt 4: Anzeigen der erstellten Nachverfolgung

In diesem Schritt interagieren Sie mit der X-Ray-Konsole, um den von der Beispielanwendung erstellten Trace anzuzeigen. Einen detaillierteren Leitfaden für die Analyse von Nachverfolgungen finden Sie unter Anzeigen der Service-Übersicht.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Zeigen Sie Segmente an, die von API Gateway, der Lambda-Funktion und dem Lambda-Container generiert wurden.
3. Sehen Sie sich unter dem Lambda-Funktionssegment ein Untersegment mit dem Namen an. `My First Serverless App` Dahinter folgt ein zweites Untersegment mit dem Namen `https://aws.amazon.com`.
4. Während der Initialisierung generiert Lambda möglicherweise auch ein drittes Untersegment mit dem Namen. `initialization`





Schritt 5: Bereinigen

Sie sollten Ressourcen, die Sie nicht mehr verwenden, stets beenden, um unerwartete Kosten zu vermeiden. Wie in diesem Tutorial gezeigt, lässt sich die serverlose erneute Bereitstellung mit Tools wie Zappa optimieren.

Um Ihre Anwendung aus Lambda, API Gateway und Amazon S3 zu entfernen, führen Sie den folgenden Befehl in Ihrem Projektverzeichnis aus, indem Sie den AWS CLI verwenden.

```
zappa undeploy dev
```

Nächste Schritte

Fügen Sie Ihrer Anwendung weitere Funktionen hinzu, indem Sie AWS Clients hinzufügen und sie mit X-Ray instrumentieren. Weitere Informationen zu den Optionen für serverloses Computing finden Sie unter Serverless on. AWS

Arbeiten mit .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Es gibt zwei Möglichkeiten, Ihre .NET-Anwendung so zu instrumentieren, dass sie Traces an X-Ray sendet:

- AWS Distro for OpenTelemetry .NET — Eine AWS Distribution, die eine Reihe von Open-Source-Bibliotheken zum Senden korrelierter Metriken und Traces über AWS Distro for Collector an mehrere AWS Überwachungslösungen wie Amazon CloudWatch und Amazon OpenSearch Service bereitstellt. AWS X-Ray OpenTelemetry
- AWS X-Ray SDK for .NET — Eine Reihe von Bibliotheken zum Generieren und Senden von Traces an X-Ray über den X-Ray-Daemon.

Weitere Informationen finden Sie unter Wahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs.

AWS Distro für .NET OpenTelemetry

Mit der AWS Distro for OpenTelemetry .NET können Sie Ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Monitoring-Lösungen wie Amazon CloudWatch AWS X-Ray, und Amazon OpenSearch Service senden. Die Verwendung von X-Ray mit AWS Distro for OpenTelemetry erfordert zwei Komponenten: ein OpenTelemetry SDK, das für die Verwendung mit X-Ray aktiviert ist, und das AWS Distro for OpenTelemetry Collector, das für die Verwendung mit X-Ray aktiviert ist.

Informationen zu den ersten Schritten finden Sie in der Dokumentation zu AWS Distro for OpenTelemetry .NET.

Weitere Informationen zur Verwendung von AWS Distro for OpenTelemetry with AWS X-Ray und anderen AWS-Services finden Sie unter AWS Distro for OpenTelemetry oder The AWS Distro for Documentation. OpenTelemetry

Weitere Informationen zur Sprachunterstützung und -verwendung finden Sie unter AWS Observability on. GitHub

AWS X-Ray SDK for .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK SDK for .NET ist eine Bibliothek für die Instrumentierung von C#.NET-Webanwendungen, .NET Core-Webanwendungen und .NET Core-Funktionen. AWS Lambda Es bietet Klassen und Methoden zum Generieren und Senden von Trace-Daten an den X-Ray-Daemon. Dazu gehören Informationen über eingehende Anfragen, die von der Anwendung bedient werden, sowie über Aufrufe, die die Anwendung an Downstream- AWS-Services, HTTP-Web APIs - und SQL-Datenbanken sendet.

> **Note**
>
> Das X-Ray SDK for .NET ist ein Open-Source-Projekt. Du kannst das Projekt verfolgen und Issues und Pull-Requests einreichen auf GitHub: github. com/aws/aws- xray-sdk-dotnet

Für Webanwendungen fügen Sie zunächst einen Message Handler zu Ihrer Web-Konfiguration hinzu, um eingehende Anforderungen zu verfolgen. Der Message Handler erstellt für jede rückverfolgte

Anforderung ein Segment und vervollständigt das Segment, nachdem die Antwort gesendet wurde. Während das Segment geöffnet ist, können Sie die SDK-Client-Methoden nutzen, um dem Segment Informationen hinzuzufügen, Untersegmente zu erstellen und nachgelagerte Aufrufe rückzuverfolgen. Das SDK erfasst auch automatisch Ausnahmen, die Ihre Anwendung ausgibt, während das Segment geöffnet ist.

Bei Lambda-Funktionen, die von einer instrumentierten Anwendung oder einem Dienst aufgerufen werden, liest Lambda den Tracing-Header und verfolgt automatisch Sampling-Anfragen. Für andere Funktionen können Sie Lambda so konfigurieren, dass eingehende Anfragen abgefragt und verfolgt werden. In beiden Fällen erstellt Lambda das Segment und stellt es dem X-Ray SDK zur Verfügung.

> **Note**
>
> Auf Lambda ist das X-Ray SDK optional. Wenn Sie es nicht in Ihrer Funktion verwenden, enthält Ihre Service-Map immer noch einen Knoten für den Lambda-Service und einen für jede Lambda-Funktion. Durch Hinzufügen des SDK können Sie Ihren Funktionscode instrumentieren, um Untersegmente zu dem von Lambda aufgezeichneten Funktionssegment hinzuzufügen. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Verwenden Sie als Nächstes das X-Ray SDK for .NET, um Ihre AWS SDK für .NET Clients zu instrumentieren. Immer wenn Sie mit einem instrumentierten Client einen Downstream AWS-Service oder eine Ressource aufrufen, zeichnet das SDK Informationen über den Anruf in einem Untersegment auf. AWS Dienste und die Ressourcen, auf die Sie innerhalb der Dienste zugreifen, werden in der Trace-Map als Downstream-Knoten angezeigt, sodass Sie Fehler und Drosselungsprobleme bei einzelnen Verbindungen leichter identifizieren können.

Das X-Ray SDK for .NET bietet auch Instrumentierung für Downstream-Aufrufe von HTTP-Web APIs - und SQL-Datenbanken. Die `GetResponseTraced`-Erweiterungsmethode für `System.Net.HttpWebRequest` verfolgt ausgehende HTTP-Aufrufe. Sie können das X-Ray SDK für die Version von .NET verwenden`SqlCommand`, um SQL-Abfragen zu instrumentieren.

Nachdem Sie das SDK verwendet haben, passen Sie sein Verhalten an, indem Sie den Rekorder und den Message-Handler konfigurieren. Sie können Plugins zum Festhalten von Daten über die Datenverarbeitungsressourcen, auf denen Ihre Anwendung ausgeführt wird, hinzufügen, das Samplingverhalten durch Samplingregeln anpassen und Protokollebenen einrichten, um mehr oder weniger Informationen von dem SDK in Ihren Anwendungsprotokollen zu sehen.

Zeichnen Sie zusätzliche Informationen zu Anforderungen und den Aufgaben, die Ihre Anwendung ausführt, in Anmerkungen und Metadaten auf. Anmerkungen sind einfache Schlüsselwertpaare, die für die Verwendung mit Filterausdrücken indiziert werden, damit Sie nach Ablaufverfolgen mit bestimmten Daten suchen können. Metadateneinträge sind weniger einschränkend und können ganze Objekte und Arrays aufzeichnen – alle Daten, die in eine JSON zusammengefasst werden können.

> **Anmerkungen und Metadaten**
>
> Anmerkungen und Metadaten sind beliebiger Text, den Sie Segmenten mit dem X-Ray SDK hinzufügen. Anmerkungen werden für die Verwendung mit Filterausdrücken indexiert. Metadaten werden nicht indexiert, können aber im Rohsegment mit der X-Ray-Konsole oder API angezeigt werden. Jeder, dem Sie Lesezugriff auf X-Ray gewähren, kann diese Daten einsehen.

Wenn Sie viele instrumentierten Clients in Ihrem Code haben, kann ein einzelnes Anforderungssegmente viele Untersegmente enthalten, eines für jeden Aufruf mit einem instrumentierten Client. Sie können Untersegmente organisieren und gruppieren, indem Sie Client-Aufrufe in benutzerdefinierten Untersegmenten zusammenfassen. Sie können ein benutzerdefiniertes Untersegment für eine ganze Funktion oder eine Code-Abschnitt erstellen und Metadaten und Anmerkungen im Untersegment festhalten, anstatt alles im übergeordneten Segment aufzuzeichnen.

Referenzdokumentation zu den SDK-Klassen und -Methoden finden Sie unter:

- AWS X-Ray SDK for .NET für.NET-API-Referenz
- AWS X-Ray SDK for .NET Core API-Referenz

Das Paket unterstützt sowohl .NET als auch .NET Core. Die verwendeten Klassen sind jedoch unterschiedlich. Beispiele in diesem Kapitel verweisen auf die .NET API-Referenz, es sei denn, die Klasse ist für .NET Core spezifisch.

Voraussetzungen

Das X-Ray SDK for .NET benötigt das.NET Framework 4.5 oder höher und AWS SDK für .NET.

Für .NET Core-Anwendungen und -Funktionen erfordert das SDK .NET Core 2.0 oder höher.

Hinzufügen des X-Ray-SDK SDK for .NET zu Ihrer Anwendung

Verwenden Sie NuGet , um das X-Ray SDK for .NET zu Ihrer Anwendung hinzuzufügen.

So installieren Sie das X-Ray SDK for .NET mit dem NuGet Paketmanager in Visual Studio

1. Wählen Sie Tools, NuGet Package Manager, Manage NuGet Packages for Solution.
2. Suchen Sie nach AWSXRayRecorder.
3. Wählen sie das Paket und dann Installieren.

Abhängigkeitsmanagement

Das X-Ray SDK for .NET ist bei Nuget erhältlich. Installieren Sie das SDK mit dem Paketmanager:

```
Install-Package AWSXRayRecorder -Version 2.10.1
```

Das `AWSXRayRecorder v2.10.1` Nuget-Paket hat die folgenden Abhängigkeiten:

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)
```

Weitere Informationen zur Abhängigkeitsverwaltung finden Sie in der Dokumentation von Microsoft zu Nuget-Abhängigkeit und Nuget-Abhängigkeitsauflösung.

Konfiguration des X-Ray SDK for .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon

auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können das X-Ray-SDK SDK for .NET mit Plug-ins so konfigurieren, dass es Informationen über den Dienst enthält, auf dem Ihre Anwendung ausgeführt wird, das standardmäßige Sampling-Verhalten ändern oder Sampling-Regeln hinzufügen, die für Anfragen an bestimmte Pfade gelten.

Für .NET-Webanwendungen fügen Sie dem Abschnitt `appSettings` Ihrer `Web.config`-Datei Schlüssel hinzu.

Example Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Erstellen Sie für .NET Core eine Datei mit dem Namen `appsettings.json` mit einem Top-Level-Schlüssel namens `XRay`.

Example.NET appsettings.json

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin",
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

Erstellen Sie dann in Ihrem Anwendungscode ein Konfigurationsobjekt und verwenden Sie es, um den X-Ray-Recorder zu initialisieren. Führen Sie dies aus, bevor Sie den Recorder initialisieren.

Example.NET Core Program.cs — Rekorder-Konfiguration

```
using Amazon.XRay.Recorder.Core;
```

```
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Wenn Sie eine .NET Core-Webanwendung instrumentieren, können Sie bei der Konfiguration des Message-Handlers auch das Konfigurationsobjekt an die `UseXRay` Methode übergeben. Verwenden Sie für Lambda-Funktionen die oben gezeigte `InitializeInstance` Methode.

Weitere Informationen zur .NET Core-Konfigurations-API finden Sie unter Konfigurieren einer ASP.NET-Core-App auf docs.microsoft.com.

Sections

- Plugins
- Regeln für die Probenahme
- Protokollierung (.NET)
- Protokollierung (.NET Core)
- Umgebungsvariablen

Plugins

Verwenden Sie Plugins zum Hinzufügen von Daten über den Service, der Ihre Anwendung hostet.

Plugins

- Amazon EC2 — `EC2Plugin` fügt die Instance-ID, die Availability Zone und die CloudWatch Logs-Gruppe hinzu.
- Elastic Beanstalk — `ElasticBeanstalkPlugin` fügt den Umgebungsnamen, die Versionsbezeichnung und die Bereitstellungs-ID hinzu.
- Amazon ECS — `ECSPlugin` fügt die Container-ID hinzu.

Um ein Plugin zu verwenden, konfigurieren Sie das X-Ray-SDK SDK for .NET .NET-Client, indem Sie die `AWSXRayPlugins` Einstellung hinzufügen. Wenn mehrere Plugins auf Ihre Anwendung zutreffen, geben Sie alle Plugins in der gleichen Einstellung an (getrennt durch Kommata).

Example Web.config – Plugins

```
<configuration>
  <appSettings>
```

```
    <add key="AWSXRayPlugins" value="EC2Plugin,ElasticBeanstalkPlugin"/>
  </appSettings>
</configuration>
```

Example.NET Core appsettings.json — Plug-ins

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin,ElasticBeanstalkPlugin"
  }
}
```

Regeln für die Probenahme

Das SDK verwendet die Sampling-Regeln, die Sie in der X-Ray-Konsole definieren, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen. Die Standardregel verfolgt die erste Anfrage jede Sekunde und fünf Prozent aller weiteren Anfragen aller Dienste, die Traces an X-Ray senden. Erstellen Sie zusätzliche Regeln in der X-Ray-Konsole, um die Menge der aufgezeichneten Daten für jede Ihrer Anwendungen anzupassen.

Das SDK wendet benutzerdefinierte Regeln in der Reihenfolge an, in der sie definiert sind. Wenn eine Anfrage mehreren benutzerdefinierten Regeln entspricht, wendet das SDK nur die erste Regel an.

> **Note**
>
> Wenn das SDK X-Ray nicht erreichen kann, um Sampling-Regeln abzurufen, kehrt es zu einer lokalen Standardregel zurück, die die erste Anfrage pro Sekunde und fünf Prozent aller zusätzlichen Anfragen pro Host vorsieht. Dies kann passieren, wenn der Host nicht berechtigt ist APIs, Sampling aufzurufen, oder wenn er keine Verbindung zum X-Ray-Daemon herstellen kann, der als TCP-Proxy für API-Aufrufe durch das SDK fungiert.

Sie können das SDK auch so konfigurieren, dass Sampling-Regeln aus einem JSON-Dokument geladen werden. Das SDK kann lokale Regeln als Backup für Fälle verwenden, in denen X-Ray Sampling nicht verfügbar ist, oder ausschließlich lokale Regeln verwenden.

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
  }
}
```

Dieses Beispiel definiert eine benutzerdefinierte Regel und eine Standardregel. Die benutzerdefinierte Regel wendet eine Stichprobenrate von fünf Prozent an, ohne dass eine Mindestanzahl von Anfragen für Pfade verfolgt werden muss. `/api/move/` Die Standardregel verfolgt die erste Anfrage jede Sekunde und 10 Prozent der weiteren Anfragen.

Der Nachteil der lokalen Definition von Regeln besteht darin, dass das feste Ziel von jeder Instanz des Rekorders unabhängig angewendet wird, anstatt vom X-Ray-Dienst verwaltet zu werden. Wenn Sie mehr Hosts bereitstellen, wird die feste Rate vervielfacht, wodurch es schwieriger wird, die Menge der aufgezeichneten Daten zu kontrollieren.

Wenn aktiviert AWS Lambda, können Sie die Samplerate nicht ändern. Wenn Ihre Funktion von einem instrumentierten Dienst aufgerufen wird, werden Aufrufe, die Anfragen generierten, die von diesem Dienst abgetastet wurden, von Lambda aufgezeichnet. Wenn aktives Tracing aktiviert ist und kein Tracing-Header vorhanden ist, trifft Lambda die Stichprobenentscheidung.

Um Backup-Regeln zu konfigurieren, weisen Sie das X-Ray SDK for .NET an, Sampling-Regeln aus einer Datei mit der `SamplingRuleManifest` Einstellung zu laden.

Example.NET Web.config – Samplingregeln

```
<configuration>
  <appSettings>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Example.NET Core appsettings.json — Sampling-Regeln

```
{
  "XRay": {
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

Um nur lokale Regeln zu verwenden, erstellen Sie den Recorder mit einer `LocalizedSamplingStrategy`. Wenn Sie Sicherungsregeln konfiguriert haben, entfernen Sie die Konfiguration.

Example.NET global.asax — Lokale Sampling-Regeln

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("samplingrules.json")).Build();
AWSXRayRecorder.InitializeInstance(recorder: recorder);
```

Example.NET Core Program.cs — Lokale Sampling-Regeln

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("sampling-rules.json")).Build();
AWSXRayRecorder.InitializeInstance(configuration,recorder);
```

Protokollierung (.NET)

Das X-Ray-SDK SDK for .NET verwendet denselben Protokollierungsmechanismus wie das AWS SDK für .NET. Wenn Sie Ihre Anwendung bereits für die Protokollierung der AWS SDK für .NET Ausgabe konfiguriert haben, gilt dieselbe Konfiguration für die Ausgabe aus dem X-Ray SDK for .NET.

Zum Konfigurieren von Protokollierung fügen Sie einen Konfigurationsabschnitt mit dem Namen `aws` Ihrer `App.config`-Datei oder `Web.config`-Datei hinzu.

Example Web.config – Protokollierung

```
...
<configuration>
  <configSections>
    <section name="aws" type="Amazon.AWSSection, AWSSDK.Core"/>
  </configSections>
```

```
  <aws>
    <logging logTo="Log4Net"/>
  </aws>
</configuration>
```

Weitere Informationen finden Sie unter Konfigurieren Ihrer AWS SDK für .NET -Anwendung im AWS SDK für .NET -Entwicklerhandbuch.

Protokollierung (.NET Core)

Das X-Ray-SDK SDK for .NET verwendet dieselben Protokollierungsoptionen wie das AWS SDK für .NET. Um die Protokollierung für .NET Core-Anwendungen zu konfigurieren, übergeben Sie die Protokollierungsoption an die `AWSXRayRecorder.RegisterLogger` Methode.

Um beispielsweise log4net zu verwenden, erstellen Sie eine Konfigurationsdatei, die den Logger, das Ausgabeformat und den Speicherort der Datei definiert.

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>
```

Erstellen Sie dann den Logger und übernehmen Sie die Konfiguration in Ihren Programmcode.

Example.NET Core Program.cs — Protokollierung

```
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)
  {
  ...
  }
}
```

Weitere Informationen zur Konfiguration von log4net finden Sie unter Konfiguration auf logging.apache.org.

Umgebungsvariablen

Sie können Umgebungsvariablen verwenden, um das X-Ray SDK for .NET zu konfigurieren. Das SDK unterstützt die folgenden Variablen.

- `AWS_XRAY_TRACING_NAME`— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet. Überschreibt den für die Segmentbenennungsstrategie des Servlet-Filters festgelegten Dienstnamen.
- `AWS_XRAY_DAEMON_ADDRESS`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest. Standardmäßig verwendet `127.0.0.1:2000` das SDK sowohl Trace-Daten (UDP) als auch Sampling-Daten (TCP). Verwenden Sie diese Variable, wenn Sie den Daemon so konfiguriert haben, dass er auf einem anderen Port lauscht oder wenn er auf einem anderen Host läuft.

  Format

  - Derselbe Port — *`address`*`:`*`port`*
  - Verschiedene Anschlüsse — `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_CONTEXT_MISSING`— Auf einstellen, `RUNTIME_ERROR` um Ausnahmen auszulösen, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, obwohl kein Segment geöffnet ist.

  Zulässige Werte

  - `RUNTIME_ERROR`— Löst eine Laufzeitausnahme aus.
  - `LOG_ERROR`— Fehler protokollieren und fortfahren (Standard).

- `IGNORE_ERROR`— Fehler ignorieren und fortfahren.

Fehler im Zusammenhang mit fehlenden Segmenten oder Untersegmenten können auftreten, wenn Sie versuchen, einen instrumentierten Client in Startcode zu verwenden, der ausgeführt wird, wenn keine Anfrage geöffnet ist, oder in Code, der einen neuen Thread erzeugt.

Instrumentierung eingehender HTTP-Anfragen mit dem X-Ray SDK for .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können das X-Ray SDK verwenden, um eingehende HTTP-Anfragen zu verfolgen, die Ihre Anwendung auf einer EC2 Instance in Amazon EC2 oder Amazon ECS bearbeitet. AWS Elastic Beanstalk

Verwenden Sie einen Meldungs-Handler, um eingehende HTTP-Anforderungen zu instrumentieren. Wenn Sie den X-Ray-Nachrichtenhandler zu Ihrer Anwendung hinzufügen, erstellt das X-Ray-SDK für.NET für jede gesampelte Anfrage ein Segment. Dieses Segment umfasst Dauer, Methode und Status der HTTP-Anforderung. Die zusätzliche Instrumentierung schafft Untersegmente zu diesem Segment.

> **Note**
>
> Für AWS Lambda Funktionen erstellt Lambda für jede abgetastete Anfrage ein Segment. Weitere Informationen finden Sie unter AWS Lambda und AWS X-Ray.

Jedes Segment hat einen Namen, der Ihre Anwendung in der Service Map identifiziert. Das Segment kann statisch benannt werden, oder Sie können das SDK so konfigurieren, dass es dynamisch auf der Grundlage des Host-Headers in der eingehenden Anfrage benannt wird. Mit der dynamischen Benennung können Sie Traces auf der Grundlage des Domainnamens in der Anfrage gruppieren und einen Standardnamen anwenden, wenn der Name nicht einem erwarteten Muster entspricht (z. B. wenn der Host-Header gefälscht ist).

> **Weitergeleitete Anfragen**
>
> Wenn ein Load Balancer oder ein anderer Vermittler eine Anfrage an Ihre Anwendung weiterleitet, nimmt X-Ray die Client-IP aus dem `X-Forwarded-For` Header in der Anfrage und nicht aus der Quell-IP im IP-Paket. Die Client-IP, die für eine weitergeleitete Anfrage aufgezeichnet wird, kann gefälscht sein und sollte daher nicht als vertrauenswürdig eingestuft werden.

Der Meldungshandler erzeugt für jede eingehende Anforderung ein Segment mit einem `http`-Block, der die folgenden Informationen enthält:

- HTTP-Methode — GET, POST, PUT, DELETE usw.
- Client-Adresse — Die IP-Adresse des Clients, der die Anfrage gesendet hat.
- Antwortcode — Der HTTP-Antwortcode für die abgeschlossene Anfrage.
- Timing — Die Startzeit (als die Anfrage empfangen wurde) und die Endzeit (als die Antwort gesendet wurde).
- Benutzeragent — Der `user-agent` aus der Anfrage.
- Länge des Inhalts — Die Länge `content-length` der Antwort.

Sections

- Instrumentierung eingehender Anforderungen (.NET)
- Instrumentierung eingehender Anforderungen (.NET Core)
- Konfiguration einer Segmentbenennungsstrategie

Instrumentierung eingehender Anforderungen (.NET)

Um die von Ihrer Anwendung verarbeiteten Anforderungen zu instrumentieren, rufen Sie in der Methode `RegisterXRay` Ihrer `Init`-Datei die Methode `global.asax` auf.

Example global.asax – Meldungshandler

```
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");
    }
  }
}
```

Instrumentierung eingehender Anforderungen (.NET Core)

Um Anfragen zu instrumentieren, die von Ihrer Anwendung bedient werden, rufen Sie die `UseXRay` Methode vor jeder anderen Middleware in der `Configure` Methode Ihrer Startup-Klasse auf, da die X-Ray-Middleware idealerweise die erste Middleware sein sollte, die die Anfrage verarbeitet, und die letzte Middleware, die die Antwort in der Pipeline verarbeitet.

> **Note**
>
> Wenn Sie für .NET Core 2.0 eine `UseExceptionHandler` Methode in der Anwendung haben, stellen Sie sicher, dass Sie nach der Methode aufrufen`UseXRay`, um sicherzustellen, dass Ausnahmen aufgezeichnet werden. `UseExceptionHandler`

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
    ...
  }
```

Die `UseXRay`-Methode kann außerdem ein Konfigurationsobjekt als zweites Argument entgegennehmen.

```
app.UseXRay("MyApp", configuration);
```

Konfiguration einer Segmentbenennungsstrategie

AWS X-Ray verwendet einen Dienstnamen, um Ihre Anwendung zu identifizieren und sie von den anderen Anwendungen, Datenbanken, externen AWS Ressourcen und Ressourcen zu unterscheiden APIs, die Ihre Anwendung verwendet. Wenn das X-Ray SDK Segmente für eingehende Anfragen generiert, zeichnet es den Dienstnamen Ihrer Anwendung im Namensfeld des Segments auf.

Das X-Ray SDK kann Segmente nach dem Hostnamen im HTTP-Anforderungsheader benennen. Dieser Header kann jedoch gefälscht sein, was zu unerwarteten Knoten in Ihrer Service Map führen kann. Um zu verhindern, dass das SDK Segmente aufgrund von Anfragen mit gefälschten Host-Headern falsch benennt, müssen Sie einen Standardnamen für eingehende Anfragen angeben.

Wenn Ihre Anwendung Anfragen für mehrere Domänen bearbeitet, können Sie das SDK so konfigurieren, dass es eine dynamische Benennungsstrategie verwendet, um dies in Segmentnamen widerzuspiegeln. Eine dynamische Benennungsstrategie ermöglicht es dem SDK, den Hostnamen für Anfragen zu verwenden, die einem erwarteten Muster entsprechen, und den Standardnamen auf Anfragen anzuwenden, bei denen dies nicht der Fall ist.

Beispielsweise können Sie über eine einzige Anwendung verfügen, die Anfragen an drei Subdomänen — `www.example.com`api.example.com, und — bedient. `static.example.com` Sie können eine dynamische Benennungsstrategie mit dem Muster verwenden`*.example.com`, um Segmente für jede Subdomain mit einem anderen Namen zu identifizieren, was zu drei Dienstknoten auf der Service-Map führt. Wenn Ihre Anwendung Anfragen mit einem Hostnamen empfängt, der nicht dem Muster entspricht, wird auf der Service Map ein vierter Knoten mit einem von Ihnen angegebenen Fallback-Namen angezeigt.

Wenn Sie denselben Namen für alle Segmente verwenden möchten, geben Sie bei der Initialisierung des Message-Handlers den Namen Ihrer Anwendung, wie im vorherigen Abschnitt gezeigt, ein. Dies hat den gleichen Effekt wie das Anlegen einer `FixedSegmentNamingStrategy` und das Übergeben an die `RegisterXRay`-Methode.

```
AWSXRayASPNET.RegisterXRay(this, new FixedSegmentNamingStrategy("MyApp"));
```

> **Note**
>
> Sie können den mit der `AWS_XRAY_TRACING_NAME`-Umgebungsvariablen in Code definierten standardmäßigen Dienstnamen überschreiben.

Eine dynamische Benennungsstrategie definiert ein Muster, dem Hostnamen entsprechen sollten, sowie einen Standardnamen, der verwendet wird, wenn der Hostname in der HTTP-Anforderung nicht mit diesem Muster übereinstimmt. Zur dynamischen Segmentbenennung erstellen Sie eine `DynamicSegmentNamingStrategy` und übergeben diese an die `RegisterXRay`-Methode.

```
AWSXRayASPNET.RegisterXRay(this, new DynamicSegmentNamingStrategy("MyApp",
 "*.example.com"));
```

Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu

OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Aufrufe tätigt, um Daten AWS-Services zu speichern, in eine Warteschlange zu schreiben oder Benachrichtigungen zu senden, verfolgt das X-Ray SDK for .NET die Aufrufe im Downstream in Untersegmenten. Verfolgte Ressourcen AWS-Services und Ressourcen, auf die Sie innerhalb dieser Services zugreifen (z. B. ein Amazon S3 S3-Bucket oder eine Amazon SQS SQS-Warteschlange), werden als Downstream-Knoten auf der Trace-Map in der X-Ray-Konsole angezeigt.

Sie können all Ihre AWS SDK für .NET Clients instrumentieren, indem Sie `RegisterXRayForAllServices` sie anrufen, bevor Sie sie erstellen.

Example SampleController.cs — DynamoDB-Client-Instrumentierung

```
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;
    });
```

Um Clients nur für einige Services zu instrumentieren, rufen Sie `RegisterXRay` statt `RegisterXRayForAllServices` auf. Ersetzen Sie den markierten Text durch den Namen der Client-Schnittstelle des Services.

```
AWSSDKHandler.RegisterXRay<IAmazonDynamoDB>()
```

Für alle Dienste können Sie den Namen der aufgerufenen API in der X-Ray-Konsole sehen. Für eine Untergruppe von Diensten fügt das X-Ray SDK dem Segment Informationen hinzu, um die Service Map detaillierter zu gestalten.

Wenn Sie beispielsweise einen Aufruf mit einem instrumentierten DynamoDB-Client tätigen, fügt das SDK den Tabellennamen dem Segment für Aufrufe hinzu, die auf eine Tabelle abzielen. In der Konsole wird jede Tabelle als separater Knoten in der Service Map angezeigt, mit einem generischen DynamoDB-Knoten für Aufrufe, die nicht auf eine Tabelle abzielen.

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements

```
{
  "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",
  }
}
```

Wenn Sie auf benannte Ressourcen zugreifen, werden durch Aufrufe der folgenden Services weitere Knoten in der Service-Übersicht erstellt. Durch Aufrufe, die keinen bestimmten Ressourcen gelten, wird ein generischer Knoten für den Service erstellt.

- Amazon DynamoDB — Tabellenname
- Amazon Simple Storage Service — Bucket und Schlüsselname
- Amazon Simple Queue Service — Name der Warteschlange

Verfolgen von Aufrufen an Downstream-HTTP-Webservices mit dem X-Ray SDK for .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Microservices oder öffentliches HTTP aufruft APIs, können Sie die `GetResponseTraced` Erweiterungsmethode des X-Ray-SDK für .NET verwenden, `System.Net.HttpWebRequest` um diese Aufrufe zu instrumentieren und die API als Downstream-Service zum Service Graph hinzuzufügen.

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();
}
```

Für asynchrone Aufrufe verwenden Sie `GetAsyncResponseTraced`.

```
request.GetAsyncResponseTraced();
```

Zeichnen Sie bei der Verwendung von `system.net.http.httpclient` Aufrufe mit dem delegierenden `HttpClientXRayTracingHandler`-Handler auf.

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);
}
```

Wenn Sie einen Aufruf einer Downstream-Web-API instrumentieren, zeichnet das X-Ray SDK for .NET ein Untersegment mit Informationen über die HTTP-Anfrage und -Antwort auf. X-Ray verwendet das Untersegment, um ein abgeleitetes Segment für die API zu generieren.

Example Untersegment für einen nachgelagerten HTTP-Aufruf

```
{
  "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 Abgeleitetes Segment für einen nachgelagerten HTTP-Anruf

```
{
  "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
}
```

Verfolgen von SQL-Abfragen mit dem X-Ray SDK for .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK SDK for .NET bietet eine Wrapper-Klasse für `System.Data.SqlClient.SqlCommand`, named, die Sie anstelle von `SqlCommand` verwenden können. Sie können einen SQL-Befehl mit der Klasse `TraceableSqlCommand` initialisieren.

Ablaufverfolgung von SQL-Abfragen mit synchronen und asynchronen Methoden

Die folgenden Beispiele zeigen, wie Sie SQL Server-Abfragen mit `TraceableSqlCommand` automatisch synchron und asynchron verfolgen.

Example **`Controller.cs`** – SQL-Client-Instrumentierung (synchron)

```
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();
  }
}
```

Sie können die Abfrage mithilfe der `ExecuteReaderAsync`-Methode asynchron ausführen.

Example **`Controller.cs`** – SQL-Client-Instrumentierung (asynchron)

```
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();
  }
}
```

Erfassen von SQL-Abfragen an SQL Server

Sie können die Erfassung von `SqlCommand.CommandText` als Teil des Untersegments aktivieren, das von Ihrer SQL-Abfrage erstellt wurde. `SqlCommand.CommandText` wird als Feld `sanitized_query` im JSON-Untersegment angezeigt. Standardmäßig ist diese Funktion aus Sicherheitsgründen deaktiviert.

> **Note**
>
> Aktivieren Sie die Sammlungsfunktion nicht, wenn Sie sensible Informationen als Klartext in Ihre SQL-Abfragen einfügen.

Sie können die Sammlung von SQL-Abfragen auf zwei Arten aktivieren:

- Legen Sie die Eigenschaft `CollectSqlQueries` in der globalen Konfiguration für Ihre Anwendung auf `true` fest.
- Legen Sie den Parameter `collectSqlQueries` in der Instance `TraceableSqlCommand` auf `true` fest, um Aufrufe innerhalb der Instance zu erfassen.

Aktivieren Sie die globale Eigenschaft CollectSqlQueries

Die folgenden Beispiele zeigen, wie Sie die Eigenschaft `CollectSqlQueries` für .NET und .NET Core aktivieren.

.NET

Um die Eigenschaft `CollectSqlQueries` in der globalen Konfiguration Ihrer Anwendung in .NET auf `true` festzulegen, ändern Sie die `appsettings` Ihrer `App.config`- oder `Web.config`-Datei wie dargestellt.

Example **`App.config`**Oder **`Web.config`** — Aktiviert die Erfassung von SQL-Abfragen global

```
<configuration>
<appSettings>
    <add key="CollectSqlQueries" value="true">
</appSettings>
</configuration>
```

.NET Core

Um die `CollectSqlQueries` Eigenschaft `true` in der globalen Konfiguration Ihrer Anwendung in .NET Core auf zu setzen, ändern Sie Ihre `appsettings.json` Datei unter dem X-Ray-Schlüssel, wie hier gezeigt.

Example **`appsettings.json`**— Aktiviert die Erfassung von SQL-Abfragen global

```
{
  "XRay": {
    "CollectSqlQueries":"true"
  }
}
```

Aktivieren Sie den collectSqlQueries Parameter

Sie können den Parameter `collectSqlQueries` in der Instance `TraceableSqlCommand` auf `true` setzen, um den SQL-Abfragetext für SQL Server-Abfragen zu erfassen, die mit dieser Instance durchgeführt wurden. Wenn Sie den Parameter auf `false` setzen, wird die Funktion `CollectSqlQuery` für die Instance `TraceableSqlCommand` deaktiviert.

> **Note**
>
> Der Wert von `collectSqlQueries` in der Instance `TraceableSqlCommand` überschreibt den in der globalen Konfiguration der Eigenschaft `CollectSqlQueries` festgelegten Wert.

Example Beispiel **`Controller.cs`** — Aktivieren Sie die Erfassung von SQL-Abfragen für die Instanz

```
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();
  }
}
```

Erstellen zusätzlicher Untersegmente

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Untersegmente erweitern das Segment eines Traces um Details über die Arbeit, die zur Bearbeitung einer Anfrage geleistet wurde. Jedes Mal, wenn Sie einen Anruf mit einem instrumentierten Client tätigen, zeichnet das X-Ray-SDK die in einem Untersegment generierten Informationen auf. Sie können zusätzliche Untersegmente erstellen, um andere Untersegmente zu gruppieren, die Leistung eines Codeabschnitts zu messen oder Anmerkungen und Metadaten aufzuzeichnen.

Um Untersegmente zu verwalten, verwenden Sie die Methoden `BeginSubsegment` und `EndSubsegment`. Führen Sie die gewünschte Bearbeitung im Untersegment in einem `try`-Block aus und verwenden Sie `AddException`, um Ausnahmen zu verfolgen. Rufen Sie `EndSubsegment` in einem `finally`-Block auf, um sicherzustellen, dass das Untersegment geschlossen ist.

Example Controller.cs — Benutzerdefiniertes Untersegment

```
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
try
{
  DoWork();
}
catch (Exception e)
{
  AWSXRayRecorder.Instance.AddException(e);
}
finally
{
  AWSXRayRecorder.Instance.EndSubsegment();
}
```

Wenn Sie ein Untersegment innerhalb eines Segments oder eines anderen Untersegments erstellen, generiert das X-Ray SDK for .NET eine ID dafür und zeichnet die Start- und Endzeit auf.

Example Untersegment mit Metadaten

```
"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"
    }
  },
```

Hinzufügen von Anmerkungen und Metadaten zu Segmenten mit dem X-Ray SDK for .NET

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können zusätzliche Informationen über Anfragen, die Umgebung oder Ihre Anwendung mit Anmerkungen und Metadaten aufzeichnen. Sie können Anmerkungen und Metadaten zu den Segmenten hinzufügen, die das X-Ray SDK erstellt, oder zu benutzerdefinierten Untersegmenten, die Sie erstellen.

Anmerkungen sind Schlüssel-Wert-Paare mit Zeichenfolgen-, Zahlen- oder booleschen Werten. Anmerkungen sind für die Verwendung mit Filterausdrücken indexiert. Berücksichtigen Sie Anmerkungen, um Daten zur Gruppierung von Ablaufverfolgungen in der Konsole zu verwenden, oder wenn Sie die `GetTraceSummaries`-API aufrufen.

Metadaten sind Schlüssel-Wert-Paare, die Werte beliebigen Typs enthalten können, einschließlich Objekte und Listen, aber nicht für die Verwendung mit Filterausdrücken indexiert sind. Verwenden Sie Metadaten, um zusätzliche Daten aufzuzeichnen, die Sie im Trace speichern möchten, aber nicht für die Suche verwenden müssen.

Sections

Anmerkungen mit dem X-Ray SDK for .NET aufnehmen

Verwenden Sie Anmerkungen, um Informationen zu Segmenten oder Untersegmenten, die zur Suche indiziert werden sollten, aufzuzeichnen.

Folgendes ist für alle Anmerkungen in X-Ray erforderlich:

Anmerkung zu Anforderungen

- Schlüssel — Der Schlüssel für eine X-Ray-Anmerkung kann bis zu 500 alphanumerische Zeichen enthalten. Sie können keine anderen Leerzeichen oder Symbole als einen Punkt oder Punkt (.) verwenden
- Werte — Der Wert für eine X-Ray-Anmerkung kann bis zu 1.000 Unicode-Zeichen enthalten.
- Die Anzahl der Anmerkungen — Sie können bis zu 50 Anmerkungen pro Spur verwenden.

Um Anmerkungen außerhalb einer Funktion aufzuzeichnen AWS Lambda

1. Rufen Sie eine Instance von `AWSXRayRecorder` ab.

   ```
   using Amazon.XRay.Recorder.Core;
   ...
   AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
   ```

2. Rufen Sie `addAnnotation` mit einem Zeichenfolgenschlüssel und einem booleschen, Int32-, Int64-, Double- oder Zeichenfolgenwert auf.

   ```
   recorder.AddAnnotation("mykey", "my value");
   ```

Das folgende Beispiel zeigt, wie Sie `putAnnotation` mit einem String-Schlüssel aufrufen, der einen Punkt und einen booleschen Wert, eine Zahl oder einen String-Wert enthält.

```
document.putAnnotation("testkey.test", "my value");
```

Um Anmerkungen innerhalb einer Funktion aufzuzeichnen AWS Lambda

Sowohl Segmente als auch Untersegmente innerhalb einer Lambda-Funktion werden von der Lambda-Laufzeitumgebung verwaltet. Wenn Sie einem Segment oder Untersegment innerhalb einer Lambda-Funktion eine Anmerkung hinzufügen möchten, müssen Sie wie folgt vorgehen:

1. Erstellen Sie das Segment oder Untersegment in der Lambda-Funktion.
2. Fügen Sie die Anmerkung dem Segment oder Untersegment hinzu.
3. Beenden Sie das Segment oder Untersegment.

Das folgende Codebeispiel zeigt Ihnen, wie Sie einem Untersegment innerhalb einer Lambda-Funktion eine Anmerkung hinzufügen:

```
#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();
}
```

Das X-Ray SDK zeichnet Anmerkungen als Schlüssel-Wert-Paare in einem `annotations` Objekt im Segmentdokument auf. Durch `addAnnotation` zweimaliges Aufrufen der Operation mit

demselben Schlüssel wird ein zuvor aufgezeichneter Wert für dasselbe Segment oder Untersegment überschrieben.

Nutzen Sie das `annotation[key]`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen durch Anmerkungen mit bestimmten Werten zu finden.

Metadaten mit dem X-Ray SDK for .NET aufzeichnen

Verwenden Sie Metadaten, um Informationen zu Segmenten oder Untersegmenten aufzuzeichnen, die Sie für die Verwendung in einer Suche nicht indizieren müssen. Bei Metadatenwerten kann es sich um Zeichenketten, Zahlen, Boolesche Werte oder jedes andere Objekt handeln, das in ein JSON-Objekt oder -Array serialisiert werden kann.

So zeichnen Sie Metadaten auf

1. Rufen Sie eine Instanz von ab`AWSXRayRecorder`, wie im folgenden Codebeispiel gezeigt:

   ```
   using Amazon.XRay.Recorder.Core;
   ...
   AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
   ```

2. Rufen Sie `AddMetadata` mit einem Zeichenfolgen-Namespace, einem Zeichenkettenschlüssel und einem Objektwert auf, wie im folgenden Codebeispiel gezeigt:

   ```
   recorder.AddMetadata("my namespace", "my key", "my value");
   ```

   Sie können den `AddMetadata` Vorgang auch nur mit einem Schlüssel- und Wertepaar aufrufen, wie im folgenden Codebeispiel gezeigt:

   ```
   recorder.AddMetadata("my key", "my value");
   ```

Wenn Sie keinen Wert für den Namespace angeben, verwendet `default` das X-Ray SDK.
Wenn Sie den `AddMetadata` Vorgang zweimal mit demselben Schlüssel aufrufen, wird ein zuvor aufgezeichneter Wert für dasselbe Segment oder Untersegment überschrieben.

Arbeiten mit Ruby

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Es gibt zwei Möglichkeiten, Ihre Ruby-Anwendung so zu instrumentieren, dass sie Traces an X-Ray sendet:

- AWS Distro for OpenTelemetry Ruby — Eine AWS Distribution, die eine Reihe von Open-Source-Bibliotheken zum Senden korrelierter Metriken und Traces über AWS Distro for Collector an mehrere AWS Überwachungslösungen CloudWatch AWS X-Ray, darunter Amazon und Amazon OpenSearch Service, bereitstellt. OpenTelemetry
- AWS X-Ray SDK for Ruby — Eine Reihe von Bibliotheken zum Generieren und Senden von Traces an X-Ray über den X-Ray-Daemon.

Weitere Informationen finden Sie unter Wahl zwischen AWS Distro for OpenTelemetry und X-Ray SDKs.

AWS Distro für Ruby OpenTelemetry

Mit der AWS Distribution für OpenTelemetry (ADOT) Ruby können Sie Ihre Anwendungen einmal instrumentieren und korrelierte Metriken und Traces an mehrere AWS Monitoring-Lösungen wie Amazon und Amazon CloudWatch Service AWS X-Ray senden. OpenSearch Die Verwendung von X-Ray mit ADOT erfordert zwei Komponenten: ein OpenTelemetry SDK, das für die Verwendung mit X-Ray aktiviert ist, und das AWS Distro for OpenTelemetry Collector, das für die Verwendung mit X-Ray aktiviert ist.

Informationen zu den ersten Schritten finden Sie in der Dokumentation zu AWS Distro for OpenTelemetry Ruby.

Weitere Informationen zur Verwendung von AWS Distro for OpenTelemetry with AWS X-Ray und anderen AWS-Services finden Sie unter AWS Distro for OpenTelemetry oder Distro for Documentation AWS . OpenTelemetry

Weitere Informationen zur Sprachunterstützung und -verwendung finden Sie unter AWS Observability on. GitHub

AWS X-Ray SDK for Ruby

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray SDK ist eine Bibliothek für Ruby-Webanwendungen, die Klassen und Methoden zum Generieren und Senden von Trace-Daten an den X-Ray-Daemon bereitstellt. Zu den Trace-Daten gehören Informationen über eingehende HTTP-Anfragen, die von der Anwendung bedient werden, sowie über Aufrufe, die die Anwendung mithilfe des AWS SDK, HTTP-Clients oder eines Active-Record-Clients an nachgeschaltete Dienste sendet. Sie können Segmente auch manuell erstellen und Debug-Informationen Anmerkungen und Metadaten hinzufügen.

Sie können das SDK herunterladen, indem Sie es Ihrer Gemfile-Datei hinzufügen und `bundle install` ausführen.

Example Gemfile

```
gem 'aws-sdk'
```

Wenn Sie Rails verwenden, fügen Sie zunächst die X-Ray SDK-Middleware hinzu, um eingehende Anfragen zu verfolgen. Ein Anforderungsfilter erstellt ein Segment. Während das Segment geöffnet

ist, können Sie die SDK-Client-Methoden nutzen, um dem Segment Informationen hinzuzufügen, Untersegmente zu erstellen und nachgelagerte Aufrufe rückzuverfolgen. Das SDK erfasst auch automatisch Ausnahmen, die Ihre Anwendung ausgibt, während das Segment geöffnet ist. Bei anderen Anwendungen als Rails können Sie Segmente manuell erstellen.

Verwenden Sie als Nächstes das X-Ray-SDK AWS SDK für Ruby, um Ihre HTTP- und SQL-Clients zu instrumentieren, indem Sie den Rekorder so konfigurieren, dass er die zugehörigen Bibliotheken patcht. Immer wenn Sie mit einem instrumentierten Client einen Downstream AWS-Service oder eine Ressource aufrufen, zeichnet das SDK Informationen über den Aufruf in einem Untersegment auf. AWS-Services und die Ressourcen, auf die Sie innerhalb der Services zugreifen, werden in der Trace-Map als Downstream-Knoten angezeigt, sodass Sie Fehler und Drosselungsprobleme bei einzelnen Verbindungen leichter identifizieren können.

Sobald Sie die ersten Schritte mit dem SDK gemacht haben, können Sie das Verhalten durch die Konfiguration des Recorders individualisieren. Sie können Plugins zum Festhalten von Daten über die Datenverarbeitungsressourcen, auf denen Ihre Anwendung ausgeführt wird, hinzufügen, das Samplingverhalten durch Samplingregeln anpassen und einen Logger bereitstellen, um mehr oder weniger Informationen von dem SDK in Ihren Anwendungsprotokollen zu sehen.

Zeichnen Sie zusätzliche Informationen zu Anforderungen und den Aufgaben, die Ihre Anwendung ausführt, in Anmerkungen und Metadaten auf. Anmerkungen sind einfache Schlüsselwertpaare, die für die Verwendung mit Filterausdrücken indiziert werden, damit Sie nach Ablaufverfolgen mit bestimmten Daten suchen können. Metadateneinträge sind weniger einschränkend und können ganze Objekte und Arrays aufzeichnen – alle Daten, die in eine JSON zusammengefasst werden können.

> **Anmerkungen und Metadaten**
>
> Anmerkungen und Metadaten sind beliebiger Text, den Sie Segmenten mit dem X-Ray SDK hinzufügen. Anmerkungen werden für die Verwendung mit Filterausdrücken indexiert. Metadaten werden nicht indexiert, können aber im Rohsegment mit der X-Ray-Konsole oder API angezeigt werden. Jeder, dem Sie Lesezugriff auf X-Ray gewähren, kann diese Daten einsehen.

Wenn Sie viele instrumentierten Clients in Ihrem Code haben, kann ein einzelnes Anforderungssegmente viele Untersegmente enthalten, eines für jeden Aufruf mit einem instrumentierten Client. Sie können Untersegmente organisieren und gruppieren, indem Sie Client-

Aufrufe in benutzerdefinierten Untersegmenten zusammenfassen. Sie können ein benutzerdefiniertes Untersegment für eine ganze Funktion oder eine Code-Abschnitt erstellen und Metadaten und Anmerkungen im Untersegment festhalten, anstatt alles im übergeordneten Segment aufzuzeichnen.

Eine Referenzdokumentation zu den Klassen und Methoden des SDK finden Sie in der API-Referenz zum AWS X-Ray SDK for Ruby.

Voraussetzungen

Das X-Ray SDK benötigt Ruby 2.3 oder höher und ist mit den folgenden Bibliotheken kompatibel:

- AWS SDK für Ruby Version 3.0 oder höher
- Rails Version 5.1 oder höher

Konfiguration des X-Ray-SDK SDK for Ruby

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Das X-Ray-SDK SDK for Ruby hat eine Klasse namens`XRay.recorder`, die den globalen Rekorder bereitstellt. Sie können die globale Aufzeichnung so konfigurieren, dass die Middleware, die Segmente für eingehende HTTP-Aufrufe erstellt, angepasst wird.

Sections

- Service-Plugins
- Regeln für die Probenahme
- Protokollierung
- Konfiguration des Recorders im Code

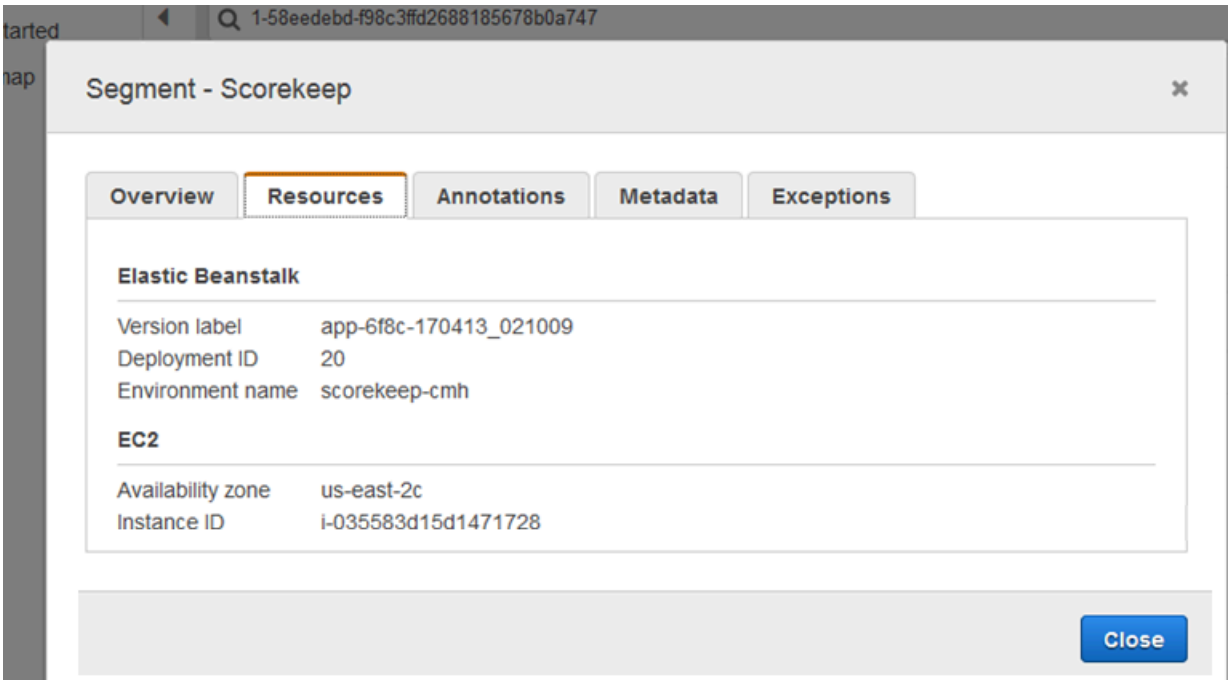
- Konfigurieren des Recorders mit Rails
- Umgebungsvariablen

Service-Plugins

Wird verwendet`plugins`, um Informationen über den Dienst aufzuzeichnen, der Ihre Anwendung hostet.

Plugins

- Amazon EC2 — `ec2` fügt die Instance-ID und die Availability Zone hinzu.
- Elastic Beanstalk — `elastic_beanstalk` fügt den Umgebungsnamen, die Versionsbezeichnung und die Bereitstellungs-ID hinzu.
- Amazon ECS — `ecs` fügt die Container-ID hinzu.



Um Plugins verwenden zu können, geben Sie sie in dem Konfigurationsobjekt an, das sie dem Recorder übergeben.

Example main.rb — Plugin-Konfiguration

```
my_plugins = %I[ec2 elastic_beanstalk]
```

```
config = {
  plugins: my_plugins,
  name: 'my app',
}

XRay.recorder.configure(config)
```

Sie können auch Umgebungsvariablen, die Vorrang über Werte im Code haben, zur Konfiguration des Recorders verwenden.

Das SDK verwendet auch Plugin-Einstellungen, um das `origin` Feld im Segment festzulegen. Dies gibt den AWS Ressourcentyp an, auf dem Ihre Anwendung ausgeführt wird. Wenn Sie mehrere Plugins verwenden, verwendet das SDK die folgende Auflösungsreihenfolge, um den Ursprung zu bestimmen: ElasticBeanstalk > EKS > ECS > EC2.

Regeln für die Probenahme

Das SDK verwendet die Sampling-Regeln, die Sie in der X-Ray-Konsole definieren, um zu bestimmen, welche Anfragen aufgezeichnet werden sollen. Die Standardregel verfolgt die erste Anfrage jede Sekunde und fünf Prozent aller weiteren Anfragen aller Dienste, die Traces an X-Ray senden. Erstellen Sie zusätzliche Regeln in der X-Ray-Konsole, um die Menge der aufgezeichneten Daten für jede Ihrer Anwendungen anzupassen.

Das SDK wendet benutzerdefinierte Regeln in der Reihenfolge an, in der sie definiert sind. Wenn eine Anfrage mehreren benutzerdefinierten Regeln entspricht, wendet das SDK nur die erste Regel an.

> **Note**
>
> Wenn das SDK X-Ray nicht erreichen kann, um Sampling-Regeln abzurufen, kehrt es zu einer lokalen Standardregel zurück, die die erste Anfrage pro Sekunde und fünf Prozent aller zusätzlichen Anfragen pro Host vorsieht. Dies kann passieren, wenn der Host nicht berechtigt ist APIs, Sampling aufzurufen, oder wenn er keine Verbindung zum X-Ray-Daemon herstellen kann, der als TCP-Proxy für API-Aufrufe durch das SDK fungiert.

Sie können das SDK auch so konfigurieren, dass Sampling-Regeln aus einem JSON-Dokument geladen werden. Das SDK kann lokale Regeln als Backup für Fälle verwenden, in denen X-Ray Sampling nicht verfügbar ist, oder ausschließlich lokale Regeln verwenden.

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
  }
}
```

In diesem Beispiel werden eine benutzerdefinierte Regel und eine Standardregel definiert. Die benutzerdefinierte Regel wendet eine Stichprobenrate von fünf Prozent an, ohne dass eine Mindestanzahl von Anfragen für Pfade verfolgt werden muss, unter `/api/move/` denen Pfade verfolgt werden müssen. Die Standardregel verfolgt die erste Anfrage jede Sekunde und 10 Prozent der weiteren Anfragen.

Der Nachteil der lokalen Definition von Regeln besteht darin, dass das feste Ziel von jeder Instanz des Rekorders unabhängig angewendet wird, anstatt vom X-Ray-Dienst verwaltet zu werden. Wenn Sie mehr Hosts bereitstellen, wird die feste Rate vervielfacht, wodurch es schwieriger wird, die Menge der aufgezeichneten Daten zu kontrollieren.

Um Sicherungsregeln zu konfigurieren, definieren Sie einen Hash für das Dokument im Konfigurationsobjekt, das Sie dem Recorder übergeben.

Example main.rb — Konfiguration der Backup-Regeln

```
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)
```

Speichern Sie die Samplingregeln unabhängig voneinander, definieren Sie den Hash in einer separaten Datei und fordern Sie die Datei an, damit er in Ihre Anwendung übernommen wird.

Example config/sampling-rules.rb

```
my_sampling_rules =  {
  version: 1,
  default: {
    fixed_target: 1,
    rate: 0.1
  }
}
```

Example main.rb — Sampling-Regel aus einer Datei

```
require 'aws-xray-sdk'
require 'config/sampling-rules.rb'

config = {
  sampling_rules: my_sampling_rules,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Um nur lokale Regeln zu verwenden, fordern Sie die Samplingregeln an und konfigurieren den `LocalSampler`.

Example main.rb — Sampling mit lokalen Regeln

```
require 'aws-xray-sdk'
require 'aws-xray-sdk/sampling/local/sampler'

config = {
  sampler: LocalSampler.new,
```

```
  name: 'my app',
}
XRay.recorder.configure(config)
```

Sie können auch die globale Aufzeichnung so konfigurieren, dass das Sampling deaktiviert und alle eingehenden Anfragen instrumentiert werden.

Example main.rb — Deaktiviert das Sampling

```
require 'aws-xray-sdk'
config = {
  sampling: false,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Protokollierung

Standardmäßig gibt der Recorder Ereignisse auf Informationsebene in `$stdout` aus. Sie können die Protokollierung anpassen, indem Sie einen Logger in dem Konfigurationsobjekt definieren, das Sie dem Recorder übergeben.

Example main.rb — Protokollierung

```
require 'aws-xray-sdk'
config = {
  logger: my_logger,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Verwenden Sie Debug-Protokolle, um Probleme wie nicht geschlossene Untersegmente zu identifizieren, wenn Sie Untersegmente manuell generieren.

Konfiguration des Recorders im Code

Zusätzliche Einstellungen finden Sie in der `configure`-Methode auf `XRay.recorder`.

- `context_missing`— Auf einstellen, um `LOG_ERROR` zu verhindern, dass Ausnahmen ausgelöst werden, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, wenn kein Segment geöffnet ist.

- `daemon_address`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest.
- `name`— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet.
- `naming_pattern`— Legen Sie ein Domainnamenmuster fest, um dynamische Benennung zu verwenden.
- `plugins`— Erfassen Sie Informationen über die AWS Ressourcen Ihrer Anwendung mit Plugins.
- `sampling`— Auf einstellen, `false` um das Sampling zu deaktivieren.
- `sampling_rules`— Legen Sie den Hash fest, der Ihre Sampling-Regeln enthält.

Example main.rb — Deaktiviert im Kontext fehlende Ausnahmen

```
require 'aws-xray-sdk'
config = {
  context_missing: 'LOG_ERROR'
}

XRay.recorder.configure(config)
```

Konfigurieren des Recorders mit Rails

Wenn Sie das Rails-Framework verwenden, können Sie die Optionen für den globalen Recorder in einer Ruby-Datei unter `app_root/initializers` konfigurieren. Das X-Ray SDK unterstützt einen zusätzlichen Konfigurationsschlüssel für die Verwendung mit Rails.

- `active_record`— Auf einstellen, `true` um Untersegmente für Active Record-Datenbanktransaktionen aufzuzeichnen.

Konfigurieren Sie die verfügbaren Einstellungen in einem Konfigurationsobjekt mit dem Namen `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
}
```

Umgebungsvariablen

Sie können Umgebungsvariablen verwenden, um das X-Ray SDK for Ruby zu konfigurieren. Das SDK unterstützt die folgenden Variablen:

- `AWS_XRAY_TRACING_NAME`— Legen Sie einen Dienstnamen fest, den das SDK für Segmente verwendet. Überschreibt den für die Segmentbenennungsstrategie des Servlet-Filters festgelegten Dienstnamen.
- `AWS_XRAY_DAEMON_ADDRESS`— Legt den Host und den Port des X-Ray-Daemon-Listeners fest. Standardmäßig sendet das SDK Trace-Daten an. `127.0.0.1:2000` Verwenden Sie diese Variable, wenn Sie den Daemon so konfiguriert haben, dass er auf einem anderen Port lauscht, oder wenn er auf einem anderen Host läuft.
- `AWS_XRAY_CONTEXT_MISSING`— Legt fest, `RUNTIME_ERROR` dass Ausnahmen ausgelöst werden, wenn Ihr instrumentierter Code versucht, Daten aufzuzeichnen, obwohl kein Segment geöffnet ist.

  Zulässige Werte

  - `RUNTIME_ERROR`— Löst eine Laufzeitausnahme aus.
  - `LOG_ERROR`— Einen Fehler protokollieren und fortfahren (Standard).
  - `IGNORE_ERROR`— Fehler ignorieren und fortfahren.

  Fehler im Zusammenhang mit fehlenden Segmenten oder Untersegmenten können auftreten, wenn Sie versuchen, einen instrumentierten Client in Startcode zu verwenden, der ausgeführt wird, wenn keine Anfrage geöffnet ist, oder in Code, der einen neuen Thread erzeugt.

Umgebungsvariablen überschreiben Werte im Code.

Nachverfolgung eingehender Anfragen mit dem X-Ray-SDK SDK for Ruby Ruby-Middleware

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon

Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können das X-Ray SDK verwenden, um eingehende HTTP-Anfragen zu verfolgen, die Ihre Anwendung auf einer EC2 Instance in Amazon EC2 oder Amazon ECS bearbeitet. AWS Elastic Beanstalk

Wenn Sie Rails einsetzen, verwenden Sie die Rails-Middleware, um eingehenden HTTP-Anforderungen zu instrumentieren. Wenn Sie die Middleware zu Ihrer Anwendung hinzufügen und einen Segmentnamen konfigurieren, erstellt das X-Ray SDK for Ruby für jede gesampelte Anfrage ein Segment. Alle durch eine zusätzliche Instrumentierung erstellten Segmente werden zu Untersegmenten des Segments auf Anfrageebene, das Informationen über die HTTP-Anforderung und Antwort bereitstellt. Diese Informationen umfassen Dauer, Methode und Status der Anfrage.

Jedes Segment hat einen Namen, der Ihre Anwendung in der Service Map identifiziert. Das Segment kann statisch benannt werden, oder Sie können das SDK so konfigurieren, dass es dynamisch auf der Grundlage des Host-Headers in der eingehenden Anfrage benannt wird. Mit der dynamischen Benennung können Sie Traces auf der Grundlage des Domainnamens in der Anfrage gruppieren und einen Standardnamen anwenden, wenn der Name nicht einem erwarteten Muster entspricht (z. B. wenn der Host-Header gefälscht ist).

Weitergeleitete Anfragen

Wenn ein Load Balancer oder ein anderer Vermittler eine Anfrage an Ihre Anwendung weiterleitet, nimmt X-Ray die Client-IP aus dem `X-Forwarded-For` Header in der Anfrage und nicht aus der Quell-IP im IP-Paket. Die Client-IP, die für eine weitergeleitete Anfrage aufgezeichnet wird, kann gefälscht sein und sollte daher nicht als vertrauenswürdig eingestuft werden.

Wenn eine Anfrage weitergeleitet wird, legt das SDK ein zusätzliches Feld im Segment fest, um dies anzuzeigen. Wenn das Segment das Feld enthält, das auf `x_forwarded_for` gesetzt ist`true`, wurde die Client-IP aus dem `X-Forwarded-For` Header in der HTTP-Anfrage übernommen.

Die Middleware erzeugt für jede eingehende Anfrage ein Segment mit einem `http`-Block mit folgenden Informationen:

- HTTP-Methode — GET, POST, PUT, DELETE usw.
- Client-Adresse — Die IP-Adresse des Clients, der die Anfrage gesendet hat.
- Antwortcode — Der HTTP-Antwortcode für die abgeschlossene Anfrage.
- Timing — Die Startzeit (als die Anfrage empfangen wurde) und die Endzeit (als die Antwort gesendet wurde).
- Benutzeragent — Der `user-agent` aus der Anfrage.
- Länge des Inhalts — Die Länge `content-length` der Antwort.

Verwenden der Rails-Middleware

Um die Middleware zu verwenden, aktualisieren Sie Ihre Gemfile-Datei, sodass sie das geforderte railtie enthält.

Example Gemfile – Rails

```
gem 'aws-xray-sdk', require: ['aws-xray-sdk/facets/rails/railtie']
```

Um die Middleware verwenden zu können, müssen Sie den Rekorder außerdem mit einem Namen konfigurieren, der der Anwendung in der Trace-Map entspricht.

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {
  name: 'my app'
}
```

Code manuell instrumentieren

Wenn Sie Rails nicht verwenden, erstellen Sie Segmente manuell. Sie können für jede eingehende Anfrage ein Segment erstellen oder Segmente rund um gepatchte HTTP- oder AWS SDK-Clients erstellen, um dem Rekorder Kontext zum Hinzufügen von Untersegmenten zu bieten.

```
# 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
```

Konfiguration einer Segmentbenennungsstrategie

AWS X-Ray verwendet einen Dienstnamen, um Ihre Anwendung zu identifizieren und sie von den anderen Anwendungen, Datenbanken, externen AWS Ressourcen und Ressourcen zu unterscheiden APIs, die Ihre Anwendung verwendet. Wenn das X-Ray SDK Segmente für eingehende Anfragen generiert, zeichnet es den Dienstnamen Ihrer Anwendung im Namensfeld des Segments auf.

Das X-Ray SDK kann Segmente nach dem Hostnamen im HTTP-Anforderungsheader benennen. Dieser Header kann jedoch gefälscht sein, was zu unerwarteten Knoten in Ihrer Service Map führen kann. Um zu verhindern, dass das SDK Segmente aufgrund von Anfragen mit gefälschten Host-Headern falsch benennt, müssen Sie einen Standardnamen für eingehende Anfragen angeben.

Wenn Ihre Anwendung Anfragen für mehrere Domänen bearbeitet, können Sie das SDK so konfigurieren, dass es eine dynamische Benennungsstrategie verwendet, um dies in Segmentnamen widerzuspiegeln. Eine dynamische Benennungsstrategie ermöglicht es dem SDK, den Hostnamen für Anfragen zu verwenden, die einem erwarteten Muster entsprechen, und den Standardnamen auf Anfragen anzuwenden, bei denen dies nicht der Fall ist.

Beispielsweise könnten Sie eine einzige Anwendung haben, die Anfragen an drei Subdomänen — `www.example.com`api.example.com, und — bedient. `static.example.com` Sie können eine dynamische Benennungsstrategie mit dem Muster verwenden`*.example.com`, um Segmente für jede Subdomain mit einem anderen Namen zu identifizieren, was zu drei Dienstknoten auf der Service-Map führt. Wenn Ihre Anwendung Anfragen mit einem Hostnamen empfängt, der nicht dem Muster entspricht, wird auf der Service Map ein vierter Knoten mit einem von Ihnen angegebenen Fallback-Namen angezeigt.

Wenn Sie denselben Namen für alle Anfragesegmente verwenden möchten, geben Sie bei der Konfiguration des Recorders den Namen Ihrer Anwendung, wie in den vorherigen Abschnitten gezeigt, ein.

Eine dynamische Benennungsstrategie definiert ein Muster, dem Hostnamen entsprechen sollten, sowie einen Standardnamen, der verwendet wird, wenn der Hostname in der HTTP-Anforderung nicht mit diesem Muster übereinstimmt. Um Segmente dynamisch zu benennen, geben Sie ein Namensmuster im Config-Hash an.

Example main.rb — Dynamische Benennung

```
config = {
  naming_pattern: '*mydomain*',
  name: 'my app',
}

XRay.recorder.configure(config)
```

Sie können "*" im Muster verwenden, um eine Übereinstimmung mit einer beliebigen Zeichenfolge zu erzielen, oder "?" für die Übereinstimmung mit einem einzelnen Zeichen.

> **Note**
>
> Sie können den mit der `AWS_XRAY_TRACING_NAME`-Umgebungsvariablen in Code definierten standardmäßigen Dienstnamen überschreiben.

Patchen von Bibliotheken zum Instrumentieren von nachgelagerten Aufrufen

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu

OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Um Downstream-Aufrufe zu instrumentieren, verwenden Sie das X-Ray-SDK SDK for Ruby, um die Bibliotheken zu patchen, die Ihre Anwendung verwendet. Das X-Ray SDK for Ruby kann die folgenden Bibliotheken patchen.

Unterstützte Bibliotheken

- `net/http`— Instrumentieren Sie HTTP-Clients.
- `aws-sdk`— AWS SDK für Ruby Instrumentenkunden.

Wenn Sie eine gepatchte Bibliothek verwenden, erstellt das X-Ray-SDK SDK for Ruby ein Untersegment für den Aufruf und zeichnet Informationen aus der Anfrage und Antwort auf. Für das SDK muss ein Segment zur Verfügung stehen, damit es ein Untersegment aus der SDK-Middleware oder einem Aufruf von `XRay.recorder.begin_segment` erstellen kann.

Um Bibliotheken zu patchen, geben Sie sie im Konfigurationsobjekt an, das Sie an den X-Ray-Recorder übergeben.

Example main.rb — Patch-Bibliotheken

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http aws_sdk]
}

XRay.recorder.configure(config)
```

Verfolgen von AWS SDK-Aufrufen mit dem X-Ray SDK for Ruby

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon

auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon
Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Wenn Ihre Anwendung Aufrufe tätigt, um Daten AWS-Services zu speichern, in eine Warteschlange zu schreiben oder Benachrichtigungen zu senden, verfolgt das X-Ray SDK for Ruby die Aufrufe im Downstream in Untersegmenten. Verfolgte Ressourcen AWS-Services und Ressourcen, auf die Sie innerhalb dieser Services zugreifen (z. B. ein Amazon S3 S3-Bucket oder eine Amazon SQS SQS-Warteschlange), werden als Downstream-Knoten auf der Trace-Map in der X-Ray-Konsole angezeigt.

Das X-Ray-SDK SDK for Ruby instrumentiert automatisch alle AWS SDK-Clients, wenn Sie die `aws-sdk` Bibliothek patchen. Einzelne Clients können nicht instrumentiert werden.

Für alle Dienste können Sie den Namen der aufgerufenen API in der X-Ray-Konsole sehen. Für eine Untergruppe von Diensten fügt das X-Ray SDK dem Segment Informationen hinzu, um die Service Map detaillierter zu gestalten.

Wenn Sie beispielsweise einen Aufruf mit einem instrumentierten DynamoDB-Client tätigen, fügt das SDK den Tabellennamen dem Segment für Aufrufe hinzu, die auf eine Tabelle abzielen. In der Konsole wird jede Tabelle als separater Knoten in der Service Map angezeigt, mit einem generischen DynamoDB-Knoten für Aufrufe, die nicht auf eine Tabelle abzielen.

Example Untersegment für einen Aufruf von DynamoDB zum Speichern eines Elements

```
{
  "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",
  }
}
```

Wenn Sie auf benannte Ressourcen zugreifen, werden durch Aufrufe der folgenden Services weitere Knoten in der Service-Übersicht erstellt. Durch Aufrufe, die keinen bestimmten Ressourcen gelten, wird ein generischer Knoten für den Service erstellt.

- Amazon DynamoDB — Tabellenname
- Amazon Simple Storage Service — Bucket und Schlüsselname
- Amazon Simple Queue Service — Name der Warteschlange

Generieren benutzerdefinierter Untersegmente mit dem X-Ray SDK

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Untersegmente erweitern das Segment eines Traces um Details über die Arbeit, die zur Bearbeitung einer Anfrage geleistet wurde. Jedes Mal, wenn Sie einen Anruf mit einem instrumentierten Client tätigen, zeichnet das X-Ray-SDK die in einem Untersegment generierten Informationen auf. Sie können zusätzliche Untersegmente erstellen, um andere Untersegmente zu gruppieren, die Leistung eines Codeabschnitts zu messen oder Anmerkungen und Metadaten aufzuzeichnen.

Um Untersegmente zu verwalten, verwenden Sie die Methoden `begin_subsegment` und `end_subsegment`.

```
subsegment = XRay.recorder.begin_subsegment name: 'annotations', namespace: 'remote'
```

```
my_annotations = { id: 12345 }
subsegment.annotations.update my_annotations
XRay.recorder.end_subsegment
```

Um ein Subsegment für eine Funktion zu erstellen, kapseln Sie es in einen Aufruf von `XRay.recorder.capture` ein.

```
XRay.recorder.capture('name_for_subsegment') do |subsegment|
  resp = myfunc() # myfunc is your function
  subsegment.annotations.update k1: 'v1'
  resp
end
```

Wenn Sie ein Untersegment innerhalb eines Segments oder eines anderen Untersegments erstellen, generiert das X-Ray SDK eine ID dafür und zeichnet die Start- und Endzeit auf.

Example Untersegment mit Metadaten

```
"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"
    }
  },
```

Hinzufügen von Anmerkungen und Metadaten zu Segmenten mit dem X-Ray SDK for Ruby

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon
> Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu

OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Sie können zusätzliche Informationen über Anfragen, die Umgebung oder Ihre Anwendung mit Anmerkungen und Metadaten aufzeichnen. Sie können Anmerkungen und Metadaten zu den Segmenten hinzufügen, die das X-Ray SDK erstellt, oder zu benutzerdefinierten Untersegmenten, die Sie erstellen.

Anmerkungen sind Schlüssel-Wert-Paare mit Zeichenfolgen-, Zahlen- oder booleschen Werten. Anmerkungen sind für die Verwendung mit Filterausdrücken indexiert. Berücksichtigen Sie Anmerkungen, um Daten zur Gruppierung von Ablaufverfolgungen in der Konsole zu verwenden, oder wenn Sie die `GetTraceSummaries`-API aufrufen.

Metadaten sind Schlüssel-Wert-Paare, die Werte beliebigen Typs enthalten können, einschließlich Objekte und Listen, aber nicht für die Verwendung mit Filterausdrücken indexiert sind. Verwenden Sie Metadaten, um zusätzliche Daten aufzuzeichnen, die Sie im Trace speichern möchten, aber nicht für die Suche verwenden müssen.

Zusätzlich zu Anmerkungen und Metadaten können Sie auch Benutzer-ID-Zeichenfolgen in Segmenten aufzeichnen. Benutzer IDs werden in einem separaten Feld in Segmenten aufgezeichnet und für die Verwendung bei der Suche indexiert.

Sections

Anmerkungen mit dem X-Ray SDK for Ruby aufnehmen

Verwenden Sie Anmerkungen, um Informationen zu Segmenten oder Untersegmenten, die zur Suche indiziert werden sollten, aufzuzeichnen.

Anmerkung zu Anforderungen

- Schlüssel — Der Schlüssel für eine X-Ray-Anmerkung kann bis zu 500 alphanumerische Zeichen enthalten. Sie können keine anderen Leerzeichen oder Symbole als einen Punkt oder Punkt (.) verwenden

- Werte — Der Wert für eine X-Ray-Anmerkung kann bis zu 1.000 Unicode-Zeichen enthalten.
- Die Anzahl der Anmerkungen — Sie können bis zu 50 Anmerkungen pro Spur verwenden.

So zeichnen Sie Anmerkungen auf

1. Eine Referenz des aktuellen Segments oder Untersegments finden Sie unter `xray_recorder`.

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_segment
   ```

   oder

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_subsegment
   ```

2. Rufen Sie `update` mit einem Hash-Wert auf.

   ```
   my_annotations = { id: 12345 }
   document.annotations.update my_annotations
   ```

   Das folgende Beispiel zeigt, wie Sie `update` mit einem Annotationsschlüssel aufrufen, der einen Punkt enthält.

   ```
   my_annotations = { testkey.test: 12345 }
   document.annotations.update my_annotations
   ```

Das SDK zeichnet Anmerkungen als Schlüssel-Wert-Paare in einem `annotations`-Objekt im Segmentdokument auf. Wenn `add_annotations` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Nutzen Sie das `annotation[`*`key`*`]`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen durch Anmerkungen mit bestimmten Werten zu finden.

Metadaten mit dem X-Ray SDK for Ruby aufnehmen

Verwenden Sie Metadaten, um Segment- oder Untersegmentinformationen aufzuzeichnen, die nicht zur Suche indiziert werden müssen. Metadatenwerte sind Zeichenfolgen, Zahlen, boolesche Werte oder andere Objekte, die in Form eines JSON-Objekts oder eines Arrays angeordnet sein können.

So zeichnen Sie Metadaten auf

1. Eine Referenz des aktuellen Segments oder Untersegments finden Sie unter `xray_recorder`.

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_segment
   ```

   oder

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_subsegment
   ```

2. Rufen Sie `metadata` mit einem Zeichenfolgenschlüssel, einem booleschen Wert, einer Zahl, einer Zeichenfolge oder einem Objektwert und einem Zeichenfolgen-Namespace auf.

   ```
   my_metadata = {
     my_namespace: {
       key: 'value'
     }
   }
   subsegment.metadata my_metadata
   ```

Wenn `metadata` zweimal mit demselben Schlüssel aufgerufen wird, werden zuvor aufgezeichnete Werte im gleichen Segment oder Untersegment überschrieben.

Benutzer IDs mit dem X-Ray SDK for Ruby aufnehmen

Zeichnen Sie Segmente von Benutzern IDs auf Anfrage auf, um den Benutzer zu identifizieren, der die Anfrage gesendet hat.

Um den Benutzer aufzuzeichnen IDs

1. Eine Referenz des aktuellen Segments finden Sie unter `xray_recorder`.

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_segment
   ```

2. Setzen Sie das Benutzerfeld auf dem Segment auf eine Zeichenfolgen-ID des Benutzers, der die Anforderung gesendet hat.

   ```
   segment.user = 'U12345'
   ```

Sie können den Benutzern in Ihren Controllern festlegen, um die Benutzer-ID aufzuzeichnen, sobald die Anwendung mit der Bearbeitung einer Anfrage beginnt.

Nutzen Sie das `user`-Schlüsselwort in einem Filterausdruck, um Ablaufverfolgungen einer Benutzer-ID zu finden.

Zeitplan für die Support von X-Ray SDK und Daemon

In der folgenden Tabelle sind die Daten und der Grad der Unterstützung für X-Ray SDKs und Daemon aufgeführt.

SDK- und Daemon-Phase	Datum des Beginns	Enddatum	Support zur Verfügung gestellt
Allgemeine Verfügbarkeit	N/A	25. Februar 2026	X-Ray SDKs und Daemon werden vollständig unterstützt. AWS bietet regelmäßige SDK- und Daemon-Versionen, die Fehler- und Sicherheitskorrekturen enthalten.
Wartungsmodus	25. Februar 2026	–	AWS schränkt die Versionen von X-Ray SDK und Daemon ein, um nur Sicherheitsprobleme zu beheben. Sie SDKs/Daemon werden keine neuen Funktionserweiterungen erhalten.

Wir empfehlen Ihnen, auf OpenTelemetry Lösungen für die Instrumentierung Ihrer Anwendung und das Senden von Traces an AWS X-Ray zu migrieren. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.

Migration von Röntgeninstrumenten zur OpenTelemetry Instrumentierung

> **Note**
>
> SDK/Daemon X-Ray-Wartungshinweis — Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zum Zeitplan für den Support finden Sie unter. Zeitplan für die Support von X-Ray SDK und Daemon

X-Ray stellt auf OpenTelemetry (OTel) als primären Instrumentierungsstandard für Anwendungsverfolgung und Beobachtbarkeit um. Dieser strategische Wandel orientiert sich an den AWS Best Practices der Branche und bietet Kunden eine umfassendere, flexiblere und zukunftsfähigere Lösung für ihre Observability-Anforderungen. OpenTelemetryDie breite Akzeptanz in der Branche ermöglicht die Rückverfolgung von Anfragen über verschiedene Systeme hinweg, auch über Systeme außerhalb, AWS die möglicherweise nicht direkt in X-Ray integriert sind.

Dieses Kapitel enthält Empfehlungen für einen reibungslosen Übergang und betont, wie wichtig die Migration zu OpenTelemetry basierten Lösungen ist, um weiterhin Support und Zugriff auf die neuesten Funktionen in den Bereichen Anwendungsinstrumentierung und Beobachtbarkeit zu gewährleisten.

Es wird empfohlen, die Lösung OpenTelemetry als Observability-Lösung für die Instrumentierung Ihrer Anwendung zu verwenden.

Themen

- Migrieren Sie zu .NET OpenTelemetry
- Zu OpenTelemetry Python migrieren
- Migrieren Sie zu Ruby OpenTelemetry

Verstehen OpenTelemetry

OpenTelemetry ist ein branchenübliches Observability-Framework, das standardisierte Protokolle und Tools für die Erfassung von Telemetriedaten bereitstellt. Es bietet einen einheitlichen Ansatz für die Instrumentierung, Generierung, Erfassung und den Export von Telemetriedaten wie Metriken, Protokollen und Traces.

Wenn Sie von X-Ray SDKs zu migrieren OpenTelemetry, erhalten Sie die folgenden Vorteile:

- Verbesserte Unterstützung für Framework- und Bibliotheksinstrumentierung
- Support für zusätzliche Programmiersprachen
- Automatische Instrumentierungsfunktionen
- Flexible Konfigurationsoptionen für die Probenahme
- Einheitliche Erfassung von Metriken, Protokollen und Traces

Der OpenTelemetry Collector bietet mehr Optionen für Datenerfassungsformate und Exportziele als der X-Ray-Daemon.

OpenTelemetry Unterstützung in AWS

AWS bietet mehrere Lösungen für die Arbeit mit OpenTelemetry:

- AWS Distribution für OpenTelemetry

  Exportieren Sie OpenTelemetry Spuren als Segmente nach X-Ray.

  Weitere Informationen finden Sie unter AWS Distro for OpenTelemetry.
- CloudWatch Signale der Anwendung

  Exportieren Sie benutzerdefinierte OpenTelemetry Traces und Metriken, um den Zustand der Anwendung zu überwachen.

  Weitere Informationen finden Sie unter Mit Anwendungssignalen arbeiten.

- CloudWatch OTel Endpunkt

  Exportieren Sie OpenTelemetry Traces mithilfe des OTel HTTP-Endpunkts mit nativer OpenTelemetry Instrumentierung nach X-Ray.

  Weitere Informationen finden Sie unter OTel Endpunkte verwenden.

Verwenden mit OpenTelemetry AWS CloudWatch

AWS CloudWatch unterstützt OpenTelemetry Traces durch clientseitige Anwendungsinstrumentierung und native AWS CloudWatch Dienste wie Application Signals, Trace, Map, Metrics und Logs. Weitere Informationen finden Sie unter OpenTelemetry.

OpenTelemetry Konzepte für Migration verstehen

In der folgenden Tabelle werden die X-Ray-Konzepte ihren OpenTelemetry Entsprechungen zugeordnet. Wenn Sie diese Mappings verstehen, können Sie Ihre vorhandenen Röntgeninstrumente wie folgt übersetzen: OpenTelemetry

X-Ray-Konzept	OpenTelemetry Konzept
Röntgenrekorder	Tracer-Anbieter und Tracer
Service-Plugins	Ressourcendetektor
Segment	(Server-) Span
Untersegment	(Nicht-Server-) Span
Regeln für die Röntgenprobenahme	OpenTelemetry Probenahme (anpassbar)
Röntgenstrahler	Span Exporter (anpassbar)
Anmerkungen/Metadaten	Attribute
Instrumentierung der Bibliothek	Instrumentierung der Bibliothek
Kontext von X-Ray Trace	Kontext überspannen
Übertragung des Kontextes von Röntgenspuren	Weitergabe des W3C-Trace-Kontextes

X-Ray-Konzept	OpenTelemetry Konzept
Röntgenspurenprobenahme	OpenTelemetry Spurenprobenahme
–	Verarbeitung im Span
–	Baggage
X-Ray-Daemon	OpenTelemetry Sammler

> **Note**
>
> Weitere Informationen zu OpenTelemetry Konzepten finden Sie in der OpenTelemetry Dokumentation.

Funktionen vergleichen

Die folgende Tabelle zeigt, welche Funktionen in beiden Diensten unterstützt werden. Verwenden Sie diese Informationen, um etwaige Lücken zu identifizieren, die Sie während der Migration beheben müssen:

Feature	Röntgeninstrumentierung	OpenTelemetry Instrumentierung
Instrumentierung der Bibliothek	Unterstützt	Unterstützt
Röntgenprobenentnahme	Unterstützt	Wird in OTel Java/.net/GO unterstützt Wird in ADOT Java/ unterstützt. NET/Python/Node.js
Übertragung des Kontextes von Röntgenspuren	Unterstützt	Unterstützt
Erkennung von Ressourcen	Unterstützt	Unterstützt

Feature	Röntgeninstrumentierung	OpenTelemetry Instrumentierung
Anmerkungen segmentieren	Unterstützt	Unterstützt
Metadaten segmentieren	Unterstützt	Unterstützt
Automatische Null-Code-Instrumentierung	Wird in Java unterstützt	Wird in OTel Java/ unterstützt. NET/Python/Node.js Wird in ADOT Java/ unterstützt. NET/Python/Node.js
Manuelles Verfolgen der Erstellung	Unterstützt	Unterstützt

Tracing einrichten und konfigurieren

Um Traces in zu erstellen OpenTelemetry, benötigen Sie einen Tracer. Sie erhalten einen Tracer, indem Sie einen Tracer-Anbieter in Ihrer Anwendung initialisieren. Dies ähnelt der Verwendung des X-Recorders zur Konfiguration von X-Ray und zum Erstellen von Segmenten und Untersegmenten in einer Röntgenspur.

> **Note**
>
> Der OpenTelemetry Tracer Provider bietet mehr Konfigurationsoptionen als der X-Ray Recorder.

Die Struktur von Trace-Daten verstehen

Nachdem Sie sich mit den grundlegenden Konzepten und Feature-Mappings vertraut gemacht haben, können Sie sich mit spezifischen Implementierungsdetails wie der Struktur von Trace-Daten und der Probenahme vertraut machen.

OpenTelemetry verwendet Spans anstelle von Segmenten und Untersegmenten zur Strukturierung von Trace-Daten. Jeder Bereich umfasst die folgenden Komponenten:

- Name

- Eindeutige ID
- Start- und Endzeitstempel
- Spanischer Typ
- Spanischer Kontext
- Attribute (Schlüssel-Wert-Metadaten)
- Ereignisse (Protokolle mit Zeitstempel)
- Links zu anderen Bereichen
- Informationen zum Status
- Referenzen aus dem Elternbereich

Wenn Sie zu migrieren OpenTelemetry, werden Ihre Bereiche automatisch in X-Ray-Segmente oder Untersegmente konvertiert. Dadurch wird sichergestellt, dass Ihr vorhandenes CloudWatch Konsolenerlebnis unverändert bleibt.

Mit Span-Attributen arbeiten

Das X-Ray SDK bietet zwei Möglichkeiten, Daten zu Segmenten und Untersegmenten hinzuzufügen:

Anmerkungen

Schlüssel-Wert-Paare, die zum Filtern und Suchen indexiert werden

Metadaten

Schlüssel-Wert-Paare, die komplexe Daten enthalten, die nicht für die Suche indexiert sind

Standardmäßig werden OpenTelemetry Span-Attribute in X-Rohdaten in Metadaten umgewandelt. Um stattdessen bestimmte Attribute in Anmerkungen umzuwandeln, fügen Sie deren Schlüssel zur `aws.xray.annotations` Attributliste hinzu.

- Weitere Informationen zu OpenTelemetry Konzepten finden Sie unter Traces OpenTelemetry
- Einzelheiten zur Zuordnung von OpenTelemetry Daten zu Röntgendaten finden Sie unter OpenTelemetry Konvertierung von Röntgendatenmodellen

Erkennen von Ressourcen in Ihrer Umgebung

OpenTelemetry verwendet Resource Detectors, um Metadaten über die Ressourcen zu sammeln, die Telemetriedaten generieren. Diese Metadaten werden als Ressourcenattribute gespeichert. Eine Entität, die Telemetrie produziert, könnte beispielsweise ein Amazon ECS-Cluster oder eine EC2 Amazon-Instance sein, und die Ressourcenattribute, die von diesen Entitäten aufgezeichnet werden können, können den Amazon ECS-Cluster-ARN oder die EC2 Amazon-Instance-ID beinhalten.

- Informationen zu den unterstützten Ressourcentypen finden Sie unter Semantic OpenTelemetry Conventions für Ressourcen
- Informationen zu X-Ray-Dienst-Plugins finden Sie unter Konfiguration des X-Ray-SDK

Verwaltung von Probenahmestrategien

Trace Sampling hilft Ihnen dabei, Ihre Kosten im Griff zu behalten, indem Daten aus einer repräsentativen Teilmenge von Anfragen statt aus allen Anfragen gesammelt werden. OpenTelemetry Sowohl X-Ray als auch X-Ray unterstützen Sampling, implementieren es jedoch unterschiedlich.

> **Note**
>
> Die Probenahme von weniger als 100% der Spuren reduziert Ihre Kosten für die Beobachtbarkeit und bietet gleichzeitig aussagekräftige Einblicke in die Leistung Ihrer Anwendung.

OpenTelemetry bietet mehrere integrierte Sampling-Strategien und ermöglicht es Ihnen, benutzerdefinierte Strategien zu erstellen. Sie können in einigen SDK-Sprachen auch einen X-Ray Remote Sampler für die Verwendung von X-Ray Sampling-Regeln konfigurieren. OpenTelemetry

Die zusätzlichen Sampling-Strategien von OpenTelemetry sind:

- Stichprobenerhebung durch die Eltern — Respektiert die Entscheidung des Elternteils, bevor zusätzliche Stichprobenstrategien angewendet werden
- Stichprobennahme auf Basis des Spurkennungsverhältnisses — >Stichprobenahme nach dem Zufallsprinzip anhand eines bestimmten Prozentsatzes von Mess
- Stichprobenentnahme — Wendet Stichprobenregeln auf vollständige Spuren im Collector an OpenTelemetry

- Benutzerdefinierte Sampler — Implementieren Sie mithilfe der Sampling-Schnittstelle Ihre eigene Sampling-Logik

Informationen zu den Regeln für die Röntgenabtastung finden Sie unter Sampling-Regeln in der X-Ray-Konsole

Informationen zu OpenTelemetry Tail Sampling finden Sie unter Tail Sampling Processor

Den Trace-Kontext verwalten

X-Ray SDKs verwaltet den Segmentkontext, um übergeordnete und untergeordnete Beziehungen zwischen Segmenten und Untersegmenten in einer Spur korrekt zu behandeln. OpenTelemetry verwendet einen ähnlichen Mechanismus, um sicherzustellen, dass Bereiche den richtigen übergeordneten Bereich haben. Es speichert und verbreitet Ablaufverfolgungsdaten im gesamten Anforderungskontext. Wenn Ihre Anwendung beispielsweise eine Anfrage verarbeitet und einen Server-Span erstellt, der diese Anforderung repräsentiert, OpenTelemetry wird der Server-Span im OpenTelemetry Kontext gespeichert, sodass bei der Erstellung eines untergeordneten Bereichs dieser untergeordnete Bereich den Span im Kontext als übergeordnetes Element referenzieren kann.

Der Trace-Kontext wird weitergegeben

Sowohl X-Ray als auch HTTP-Header OpenTelemetry verwenden, um den Trace-Kontext dienstübergreifend zu verbreiten. Auf diese Weise können Sie Trace-Daten, die von verschiedenen Diensten generiert wurden, miteinander verknüpfen und Stichprobenentscheidungen treffen.

Das X-Ray SDK verbreitet automatisch den Trace-Kontext mithilfe des X-Ray-Trace-Headers. Wenn ein Dienst einen anderen aufruft, enthält der Trace-Header den Kontext, der benötigt wird, um die Beziehungen zwischen übergeordneten und untergeordneten Traces aufrechtzuerhalten.

OpenTelemetry unterstützt mehrere Trace-Header-Formate für die Kontextweiterleitung, darunter:

- W3C Trace Context (Standard)
- Röntgen-Trace-Header
- Andere benutzerdefinierte Formate

> **Note**
>
> Sie können OpenTelemetry die Verwendung eines oder mehrerer Header-Formate konfigurieren. Verwenden Sie beispielsweise den X-Ray Propagator, um Trace-Kontext an AWS Dienste zu senden, die X-Ray Tracing unterstützen.

Konfigurieren und verwenden Sie den X-Ray Propagator, um die dienstübergreifende AWS Ablaufverfolgung zu ermöglichen. Auf diese Weise können Sie den Trace-Kontext an API-Gateway-Endpunkte und andere Dienste weitergeben, die X-Ray unterstützen.

- Informationen zu X-Ray-Trace-Headern finden Sie unter Tracing-Header im X-Ray Developer Guide
- Informationen zur OpenTelemetry Kontextweiterleitung finden Sie in der Dokumentation unter Kontext und Kontextweiterleitung OpenTelemetry

Verwendung von Bibliotheksinstrumentierung

Sowohl X-Ray als auch OpenTelemetry bieten Bibliotheksinstrumente, die nur minimale Codeänderungen erfordern, um Ihren Anwendungen Tracing hinzuzufügen.

X-Ray bietet Funktionen zur Bibliotheksinstrumentierung. Auf diese Weise können Sie vorgefertigte X-Ray-Instrumente mit minimalen Änderungen am Anwendungscode hinzufügen. Diese Instrumentierungen unterstützen spezifische Bibliotheken wie das AWS SDK und die HTTP-Clients sowie Web-Frameworks wie Spring Boot oder Express.js.

OpenTelemetryDie Instrumentierungsbibliotheken generieren durch Bibliotheks-Hooks oder automatische Codeänderungen detaillierte Bereiche für Ihre Bibliotheken, sodass nur minimale Codeänderungen erforderlich sind.

Um festzustellen, ob OpenTelemetry's Library Instrumentations Ihre Bibliothek unterstützt, suchen Sie in der OpenTelemetry Registry unter OpenTelemetry Registry danach.

Spuren exportieren

X-Ray und OpenTelemetry verwenden Sie verschiedene Methoden, um Trace-Daten zu exportieren.

Export von Röntgenspuren

Das X-Ray SDKs verwendet einen Sender, um Trace-Daten zu senden:

- Sendet Segmente und Untersegmente an den X-Ray Daemon
- Verwendet UDP für nicht blockierende I/O
- Standardmäßig im SDK konfiguriert

OpenTelemetry Trace-Export

OpenTelemetry verwendet konfigurierbare Span Exporter zum Senden von Trace-Daten:

- Verwendet die Protokolle http/protobuf oder grpc
- Exportiert Spans zu Endpunkten, die vom Collector oder Agent überwacht werden OpenTelemetry CloudWatch
- Ermöglicht benutzerdefinierte Exportkonfigurationen

Bearbeitung und Weiterleitung von Spuren

Sowohl X-Ray als auch OpenTelemetry bieten Komponenten zum Empfangen, Verarbeiten und Weiterleiten von Trace-Daten.

Verarbeitung von Röntgenspuren

Der X-Ray-Daemon kümmert sich um die Trace-Verarbeitung:

- Hört auf UDP-Verkehr von X-Ray SDKs
- Stapelt Segmente und Untersegmente
- Lädt Stapel in den X-Ray-Dienst hoch

OpenTelemetry Verarbeitung von Rückverfolgungen

Der OpenTelemetry Collector übernimmt die Trace-Verarbeitung:

- Empfängt Traces von instrumentierten Diensten
- Verarbeitet und ändert optional Trace-Daten
- Sendet verarbeitete Traces an verschiedene Backends, einschließlich X-Ray

> **Note**
>
> Der AWS CloudWatch Agent kann auch OpenTelemetry Spuren empfangen und an X-Ray senden. Weitere Informationen finden Sie unter Metriken und Traces sammeln mit OpenTelemetry.

Span Processing (OpenTelemetry-spezifisches Konzept)

OpenTelemetry verwendet Span-Prozessoren, um Spans bei ihrer Erstellung zu ändern:

- Ermöglicht das Lesen und Ändern von Spans bei der Erstellung oder Fertigstellung
- Aktiviert benutzerdefinierte Logik für die Verarbeitung von Spannen

Gepäck (OpenTelemetry-spezifisches Konzept)

OpenTelemetryDie Gepäckfunktion ermöglicht die Weitergabe von Schlüsselwertdaten:

- Ermöglicht die Übergabe beliebiger Daten zusammen mit dem Trace-Kontext
- Nützlich für die Weitergabe anwendungsspezifischer Informationen über Dienstgrenzen hinweg

Informationen zum Collector finden Sie unter OpenTelemetry Collector OpenTelemetry

Informationen zu X-Ray-Konzepten finden Sie unter X-Ray-Konzepte im X-Ray-Entwicklerhandbuch

Überblick über die Migration

Dieser Abschnitt bietet einen Überblick über die Codeänderungen, die für die Migration erforderlich sind. Die folgende Liste enthält sprachspezifische Anleitungen und Schritte zur Migration von X-Ray Daemon.

> **Important**
>
> Für eine vollständige Migration von Röntgeninstrumenten zur OpenTelemetry Instrumentierung müssen Sie:
>
> 1. Ersetzen Sie die Verwendung des X-Ray-SDK durch eine OpenTelemetry Lösung

2. Ersetzen Sie den X-Ray-Daemon durch den CloudWatch Agenten oder OpenTelemetry Collector (mit X-Ray Exporter)

- Migration zu Java OpenTelemetry
- Zu OpenTelemetry Go migrieren
- Migrieren Sie zu Node.js OpenTelemetry
- Migrieren Sie zu .NET OpenTelemetry
- Zu OpenTelemetry Python migrieren
- Migrieren Sie zu Ruby OpenTelemetry

Empfehlungen für neue und bestehende Anwendungen

Für neue und bestehende Anwendungen wird empfohlen, die folgenden Lösungen zu verwenden, um die Ablaufverfolgung in Ihren Anwendungen zu aktivieren:

Instrumentierung

- OpenTelemetry SDKs
- AWS Distribution für Instrumentierung OpenTelemetry

Datenerfassung

- OpenTelemetry Sammler
- CloudWatch Agentin

Nach der Migration zu OpenTelemetry basierten Lösungen bleibt Ihre CloudWatch Erfahrung unverändert. Sie können Ihre Traces weiterhin im gleichen Format auf den Seiten Traces und Trace Map der CloudWatch Konsole anzeigen oder Ihre Trace-Daten über X-Ray abrufen APIs.

Änderungen am Setup nachverfolgen

Sie müssen das X-Ray-Setup durch ein OpenTelemetry Setup ersetzen.

Vergleich von X-Ray und OpenTelemetry Setup

Feature	X-Ray SDK	OpenTelemetry
Standardkonfigurationen	• Zentralisierte Röntgenpr obenentnahme • Übertragung des X-Ray-Tra ce-Kon • Trace-Export nach X-Ray Daemon	• Exportieren von Traces nach OpenTelemetry Collector oder CloudWatch Agent (HTTP/gRPC) • Weitergabe des W3C-Trace -Kontextes
Manuelle Konfigurationen	• Lokale Probenahmeregeln • Plug-ins zur Ressource nerkennung	• X-Ray Sampling (mögliche rweise nicht für alle Sprachen verfügbar) • Erkennung von Ressourcen • Übertragung des X-Ray-Tra ce-Kon

Änderungen der Instrumentierung der Bibliothek

Aktualisieren Sie Ihren Code so, dass er OpenTelemetry Library Instrumentation anstelle von X-Ray Library Instrumentation für AWS SDK, HTTP-Clients, Web Frameworks und andere Bibliotheken verwendet. Dadurch werden OpenTelemetry Traces statt X-Ray Traces generiert.

> **Note**
>
> Codeänderungen variieren je nach Sprache und Bibliothek. Ausführliche Anweisungen finden Sie in den sprachspezifischen Migrationsleitfäden.

Änderungen an der Instrumentierung der Lambda-Umgebung

Um sie OpenTelemetry in Ihren Lambda-Funktionen zu verwenden, wählen Sie eine der folgenden Einrichtungsoptionen:

1. Verwenden Sie einen Lambda-Layer mit automatischer Instrumentierung:
   - (Empfohlen) CloudWatch Anwendungssignale Lambda-Schicht

> **Note**
>
> Um nur die Ablaufverfolgung zu verwenden, legen Sie die Lambda-Umgebungsvariable fest. `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`

- AWS verwalteter Lambda-Layer für ADOT

2. Manuell OpenTelemetry für Ihre Lambda-Funktion einrichten:
   - Konfiguration eines einfachen Span-Prozessors mit einem X-Ray-UDP-Span-Exporter
   - Richten Sie einen X-Ray Lambda Propagator ein

Manuelles Erstellen von Trace-Daten

Ersetzen Sie Röntgensegmente und Untersegmente durch OpenTelemetry Spans:

- Verwenden Sie einen OpenTelemetry Tracer, um Spans zu erstellen
- Hinzufügen von Attributen zu Spans (entspricht X-Ray-Metadaten und Anmerkungen)

> **Important**
>
> Wenn es an X-Ray gesendet wird:
>
> - Serverbereiche werden in X-Ray-Segmente umgewandelt
> - Andere Bereiche werden in X-Ray-Untersegmente umgewandelt
> - Attribute werden standardmäßig in Metadaten konvertiert

Um ein Attribut in eine Anmerkung zu konvertieren, fügen Sie seinen Schlüssel zur `aws.xray.annotations` Attributliste hinzu. Weitere Informationen finden Sie unter Benutzerdefinierte X-Ray-Anmerkungen aktivieren.

Migration von X-Ray Daemon zu AWS CloudWatch Agent oder Collector OpenTelemetry

Sie können entweder den CloudWatch Agenten oder den OpenTelemetry Collector verwenden, um Traces von Ihren instrumentierten Anwendungen zu empfangen und sie an X-Ray zu senden.

> **Note**
>
> Die CloudWatch Agentenversion 1.300025.0 und höher kann Traces sammeln. OpenTelemetry Wenn Sie den CloudWatch Agenten anstelle des X-Ray-Daemons verwenden, reduzieren Sie die Anzahl der Agenten, die Sie verwalten müssen. Weitere Informationen finden Sie unter Erfassung von Metriken, Protokollen und Traces mit dem CloudWatch Agenten.

Sections

Migration auf Amazon EC2 - oder lokalen Servern

> **Important**
>
> Stoppen Sie den X-Ray-Daemon-Prozess, bevor Sie den CloudWatch Agenten oder OpenTelemetry Collector verwenden, um Portkonflikte zu vermeiden.

Bestehendes X-Ray-Daemon-Setup

Den Daemon installieren

Ihre bestehende X-Ray-Daemon-Nutzung wurde mit einer der folgenden Methoden installiert:

Manuelle Installation

Laden Sie die ausführbare Datei vom X-Ray-Daemon Amazon S3 S3-Bucket herunter und führen Sie sie aus.

Automatische Installation

Verwenden Sie dieses Skript, um den Daemon zu installieren, wenn Sie eine Instanz starten:

```
#!/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
```

Konfigurieren des -Daemons

Ihre bestehende X-Ray-Daemon-Nutzung wurde wie folgt konfiguriert:

- Befehlszeilenargumente
- Konfigurationsdatei (`xray-daemon.yaml`)

Example Verwendung einer Konfigurationsdatei

```
./xray -c ~/xray-daemon.yaml
```

Ausführen des Daemons

Ihre bestehende X-Ray-Daemon-Nutzung wurde mit dem folgenden Befehl gestartet:

```
~/xray-daemon$ ./xray -o -n us-east-1
```

Den Daemon entfernen

So entfernen Sie den X-Ray-Daemon von Ihrer EC2 Amazon-Instance:

1. Stoppen Sie den Daemon-Dienst:

   ```
   systemctl stop xray
   ```

2. Löschen Sie die Konfigurationsdatei:

   ```
   rm ~/path/to/xray-daemon.yaml
   ```

3. Falls konfiguriert, entfernen Sie die Protokolldatei:

   > **Note**
   >
   > Der Speicherort der Protokolldatei hängt von Ihrer Konfiguration ab:

- Konfiguration über die Befehlszeile: `/var/log/xray-daemon.log`
- Konfigurationsdatei: Überprüfen Sie die `LogPath` Einstellung

Den CloudWatch Agenten einrichten

Den Agenten installieren

Anweisungen zur Installation finden Sie unter Installation des CloudWatch Agenten auf einem lokalen Server.

Den Agenten konfigurieren

1. Erstellen Sie eine Konfigurationsdatei, um die Trace-Erfassung zu aktivieren. Weitere Informationen finden Sie unter CloudWatch Agent-Konfigurationsdatei erstellen.
2. Richten Sie IAM-Berechtigungen ein:
   - Fügen Sie eine IAM-Rolle hinzu oder geben Sie Anmeldeinformationen für den Agenten an. Weitere Informationen finden Sie unter IAM-Rollen einrichten.
   - Stellen Sie sicher, dass die Rolle oder die Anmeldeinformationen die `xray:PutTraceSegments` Berechtigung enthalten.

Starten des -Agenten

Anweisungen zum Starten des Agenten finden Sie unter Den CloudWatch Agenten über die Befehlszeile starten.

Den OpenTelemetry Collector einrichten

Den Collector installieren

Laden Sie den OpenTelemetry Collector für Ihr Betriebssystem herunter und installieren Sie ihn. Anweisungen finden Sie unter Installation des Collectors.

Den Collector konfigurieren

Konfigurieren Sie die folgenden Komponenten in Ihrem Collector:

- aws-proxy-Erweiterung

  Für Röntgenprobenentnahmen erforderlich

- OTel Empfänger

  Sammelt Spuren aus Ihrer Anwendung

- Xray-Exporter

  Sendet Spuren an X-Ray

Example Beispiel für die Collector-Konfiguration — otel-collector-config .yaml

```
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

Konfigurieren Sie AWS Anmeldeinformationen mit der `xray:PutTraceSegments` Erlaubnis. Weitere Informationen finden Sie unter Anmeldeinformationen angeben.

Den Collector starten

Führen Sie den Collector mit Ihrer Konfigurationsdatei aus:

```
otelcol --config=otel-collector-config.yaml
```

Migration auf Amazon ECS

> **Important**
>
> Ihre Aufgabenrolle muss über die entsprechenden `xray:PutTraceSegments` Berechtigungen für jeden Collector verfügen, den Sie verwenden.
> Stoppen Sie alle vorhandenen X-Ray-Daemon-Container, bevor Sie den CloudWatch Agenten- oder OpenTelemetry Collector-Container auf demselben Host ausführen, um Portkonflikte zu vermeiden.

Den Agenten verwenden CloudWatch

1. Holen Sie sich das Docker-Image aus der Amazon ECR Public Gallery.
2. Erstellen Sie eine Konfigurationsdatei mit dem Namen: `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. Speichern Sie die Konfiguration im Systems Manager Parameter Store:

    1. Öffnen Sie https://console.aws.amazon.com/systems-manager/

2. Wählen Sie Parameter erstellen
3. Geben Sie die folgenden Werte ein:
   - Name: `/ecs/cwagent/otel-config`
   - Stufe: Standard
   - Typ: Zeichenfolge
   - Datentyp: Text
   - Wert: [Fügen Sie die cw-agent-otel .json-Konfiguration hier ein]
4. Erstellen Sie eine Aufgabendefinition im Bridge-Netzwerkmodus:

   In Ihrer Aufgabendefinition hängt die Konfiguration von dem Netzwerkmodus ab, den Sie tatsächlich nutzen. Brückennetzwerke sind die Standardeinstellung und können in Ihrer Standard-VPC verwendet werden. Stellen Sie in einem Bridge-Netzwerk die `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` Umgebungsvariable so ein, dass dem OpenTelemetry SDK der Endpunkt und der Port für den CloudWatch Agenten mitgeteilt werden. Sie sollten auch einen Link von Ihrem Anwendungscontainer zum Collector-Container erstellen, damit Traces vom OpenTelemetry SDK in Ihrer Anwendung an den Collector-Container gesendet werden können.

   Example CloudWatch Definition der Agentenaufgabe

```
{
    "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"
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Bereitstellen des CloudWatch Agenten zur Erfassung von Metriken auf EC2 Amazon-Instance-Ebene auf Amazon ECS.

Den Collector verwenden OpenTelemetry

1. Holen Sie sich das Docker-Image `otel/opentelemetry-collector-contrib` von Docker Hub.
2. Erstellen Sie eine Konfigurationsdatei `otel-collector-config.yaml` mit demselben Inhalt wie im Abschnitt EC2 Amazon-Konfiguration des Collectors angezeigt, aktualisieren Sie jedoch die Endpoints, die `0.0.0.0` anstelle von `127.0.0.1` verwendet werden sollen.
3. Um diese Konfiguration in Amazon ECS zu verwenden, können Sie die Konfiguration im Systems Manager Parameter Store speichern. Rufen Sie zunächst die Systems Manager Parameter Store-Konsole auf und wählen Sie Neuen Parameter erstellen aus. Erstellen Sie einen neuen Parameter mit den folgenden Informationen:
   - Name:/ecs/otel/config(auf diesen Namen wird in der Aufgabendefinition für den Collector verwiesen)

- Stufe: Standard
- Typ: Zeichenfolge
- Datentyp: Text
- Wert: [Fügen Sie die otel-collector-config .yaml-Konfiguration hier ein]

4. Erstellen Sie eine Aufgabendefinition für die Bereitstellung des OpenTelemetry Collectors, indem Sie als Beispiel den Bridge-Netzwerkmodus verwenden.

   In der Aufgabendefinition hängt die Konfiguration vom verwendeten Netzwerkmodus ab. Brückennetzwerke sind die Standardeinstellung und können in Ihrer Standard-VPC verwendet werden. Stellen Sie in einem Bridge-Netzwerk die `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` Umgebungsvariable so ein, dass dem OpenTelemetry SDK der Endpunkt und der Port für den OpenTelemetry Collector mitgeteilt werden. Sie sollten auch einen Link von Ihrem Anwendungscontainer zum Collector-Container erstellen, damit Traces vom OpenTelemetry SDK in Ihrer Anwendung an den Collector-Container gesendet werden können.

   Example OpenTelemetry Definition der Collector-Aufgabe

```
{
    "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"
                }
            ]
        }
    ]
}
```

Migration auf Elastic Beanstalk

> **Important**
>
> Stoppen Sie den X-Ray-Daemon-Prozess, bevor Sie den CloudWatch Agenten verwenden, um Portkonflikte zu vermeiden.

Ihre bestehende X-Ray Daemon-Integration wurde mithilfe der Elastic Beanstalk Beanstalk-Konsole oder durch die Konfiguration von X-Ray Daemon in Ihrem Anwendungsquellcode mit einer Konfigurationsdatei aktiviert.

Den Agenten verwenden CloudWatch

Konfigurieren Sie den CloudWatch Agenten auf der Amazon Linux 2-Plattform mithilfe einer `.ebextensions` Konfigurationsdatei:

1. Erstellen Sie ein Verzeichnis mit dem Namen `.ebextensions` in Ihrem Projektstamm

2. Erstellen Sie eine `cloudwatch.config` innerhalb des `.ebextensions` Verzeichnisses benannte Datei mit dem folgenden Inhalt:

```
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. Nehmen Sie das `.ebextensions` Verzeichnis bei der Bereitstellung in Ihr Anwendungsquellpaket auf

Weitere Informationen zu Elastic Beanstalk Beanstalk-Konfigurationsdateien finden Sie unter Erweiterte Umgebungsanpassung mit Konfigurationsdateien.

Migration zu Java OpenTelemetry

Dieser Abschnitt enthält Anleitungen zur Migration vom X-Ray SDK zum OpenTelemetry SDK for Java Java-Anwendungen.

Sections

- Automatische Instrumentierungslösung ohne Code
- Lösungen für die manuelle Instrumentierung mit dem SDK
- Nachverfolgung eingehender Anfragen (Spring Framework-Instrumentierung)

- AWS SDK v2-Instrumentierung
- Instrumentieren von ausgehenden HTTP-Aufrufen
- Instrumentierungsunterstützung für andere Bibliotheken
- Manuelles Erstellen von Trace-Daten
- Lambda-Instrumentierung

Automatische Instrumentierungslösung ohne Code

With X-Ray Java agent

Um den X-Ray-Java-Agenten zu aktivieren, mussten die JVM-Argumente Ihrer Anwendung geändert werden.

```
-javaagent:/path-to-disco/disco-java-agent.jar=pluginPath=/path-to-disco/disco-plugins
```

With OpenTelemetry-based Java agent

Um OpenTelemetry basierte Java-Agenten zu verwenden.

- Verwenden Sie den Java-Agenten AWS Distro for OpenTelemetry (ADOT) Auto-Instrumentation für die automatische Instrumentierung mit dem ADOT-Java-Agenten. Weitere Informationen finden Sie unter Automatische Instrumentierung für Traces und Metriken mit dem Java-Agenten. Wenn Sie nur die Ablaufverfolgung verwenden möchten, deaktivieren Sie die `OTEL_METRICS_EXPORTER=none` Umgebungsvariable, um Metriken aus dem Java-Agenten zu exportieren.

  (Optional) Sie können CloudWatch Application Signals auch aktivieren, wenn Sie Ihre Anwendungen AWS mit der automatischen ADOT-Java-Instrumentierung automatisch instrumentieren, um den aktuellen Zustand der Anwendung zu überwachen und die langfristige Anwendungsleistung zu verfolgen. Application Signals bietet eine einheitliche, anwendungsorientierte Ansicht Ihrer Anwendungen, Dienste und Abhängigkeiten und hilft bei der Überwachung und Bewertung des Anwendungszustands. Weitere Informationen finden Sie unter Application Signals.
- Verwenden Sie den OpenTelemetry Java-Agenten für die automatische Instrumentierung. Weitere Informationen finden Sie unter Zero-Code-Instrumentierung mit dem Java-Agenten.

Lösungen für die manuelle Instrumentierung mit dem SDK

Tracing setup with X-Ray SDK

Um Ihren Code mit dem X-Ray SDK for Java zu instrumentieren, musste die `AWSXRay` Klasse zunächst mit Service-Plug-ins und lokalen Sampling-Regeln konfiguriert werden. Anschließend wurde ein mitgelieferter Rekorder verwendet.

```
static { AWS XRayRecorderBuilder builder = AWS
 XRayRecorderBuilder.standard().withPlugin(new EC2Plugin()).withPlugin(new
 ECSPlugin()); AWS XRay.setGlobalRecorder(builder.build());
}
```

Tracing setup with OpenTelemetry SDK

Die folgenden Abhängigkeiten sind erforderlich.

```
<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>
```

Konfigurieren Sie das OpenTelemetry SDK, indem Sie ein Objekt instanziieren `TracerProvider` und global registrieren. `OpenTelemetrySdk` Konfigurieren Sie diese Komponenten:

- Ein OTLP Span Exporter (z. B. OtlpGrpcSpanExporter) — Erforderlich für den Export von Traces an den CloudWatch Agenten oder Collector OpenTelemetry
- Ein AWS X-Ray Propagator — Erforderlich für die Weitergabe des Trace-Kontextes an AWS Dienste, die in X-Ray integriert sind
- Ein AWS X-Ray Remote Sampler — Erforderlich, wenn Sie Anfragen anhand von Röntgenprobenregeln abtasten müssen
- Resource Detectors (zum Beispiel EcsResource oder Ec2Resource) — Erkennt Metadaten des Hosts, auf dem Ihre Anwendung ausgeführt wird

```
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();
```

Nachverfolgung eingehender Anfragen (Spring Framework-Instrumentierung)

With X-Ray SDK

Informationen zur Verwendung des X-Ray-SDK mit dem Spring-Framework zur Instrumentierung Ihrer Anwendung finden Sie unter AOP mit Spring und dem X-Ray-SDK SDK for Java. Gehen Sie wie folgt vor, um AOP in Spring zu aktivieren.

1. Konfigurieren von Spring
2. Hinzufügen eines Ablaufverfolgungsfilters zu Ihrer Anwendung
3. Kommentieren Sie Ihren Code oder implementieren Sie eine Schnittstelle
4. Aktivieren von X-Ray in Ihrer Anwendung

With OpenTelemetry SDK

OpenTelemetry bietet Instrumentierungsbibliotheken zum Sammeln von Traces für eingehende Anfragen für Spring Boot-Anwendungen. Um die Spring Boot-Instrumentierung mit minimaler Konfiguration zu aktivieren, schließen Sie die folgende Abhängigkeit ein.

```
<dependency>
           <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-spring-boot-starter</artifactId>
       </dependency>
```

Weitere Informationen zur Aktivierung und Konfiguration der Spring Boot-Instrumentation für Ihr OpenTelemetry Setup finden Sie unter OpenTelemetry Erste Schritte.

Using OpenTelemetry-based Java agents

Die empfohlene Standardmethode für die Instrumentierung von Spring Boot-Anwendungen ist die Verwendung des OpenTelemetry Java-Agenten mit Bytecode-Instrumentierung, der im Vergleich zur direkten Verwendung des SDK auch mehr out-of-the-box Instrumentierungen und Konfigurationen bietet. Informationen zu den ersten Schritten finden Sie unter. Automatische Instrumentierungslösung ohne Code

AWS SDK v2-Instrumentierung

With X-Ray SDK

Das X-Ray SDK for Java kann automatisch alle AWS SDK v2-Clients instrumentieren, wenn Sie das `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` Untermodul zu Ihrem Build hinzugefügt haben.

Um die Downstream-Client-Aufrufe einzelner Clients an AWS Dienste mit AWS SDK for Java 2.2 und höher zu instrumentieren, wurde das `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` Modul aus Ihrer Build-Konfiguration ausgeschlossen und das `aws-xray-recorder-sdk-aws-sdk-v2` Modul wurde aufgenommen. Einzelne Clients wurden instrumentiert, indem sie mit einem `TracingInterceptor` konfiguriert wurden.

```
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

Um alle AWS SDK-Clients automatisch zu instrumentieren, fügen Sie das `opentelemetry-aws-sdk-2.2-autoconfigure` Untermodul hinzu.

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-aws-sdk-2.2-autoconfigure</artifactId>
            <version>2.15.0-alpha</version>
            <scope>runtime</scope>
        </dependency>
```

Um einzelne AWS SDK-Clients zu instrumentieren, fügen Sie das `opentelemetry-aws-sdk-2.2` Untermodul hinzu.

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-aws-sdk-2.2</artifactId>
            <version>2.15.0-alpha</version>
            <scope>compile</scope>
        </dependency>
```

Registrieren Sie dann einen Interceptor, wenn Sie einen AWS SDK-Client erstellen.

```
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();
```

Instrumentieren von ausgehenden HTTP-Aufrufen

With X-Ray SDK

Um ausgehende HTTP-Anfragen mit X-Ray zu instrumentieren, HttpClient war das X-Ray SDK für Javas Version des Apache erforderlich.

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
  public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
```

With OpenTelemetry SDK

Ähnlich wie das X-Ray Java SDK OpenTelemetry stellt es eine `ApacheHttpClientTelemetry` Klasse bereit, die über eine Builder-Methode verfügt, die die Erstellung einer Instanz ermöglicht,

`HttpClientBuilder` um OpenTelemetry basierte Spans und Kontextpropagierung für Apache HttpClient bereitzustellen.

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-apache-httpclient-5.2</artifactId>
            <version>2.15.0-alpha</version>
            <scope>compile</scope>
        </dependency>
```

Im Folgenden finden Sie ein Codebeispiel aus dem opentelemetry-java-instrumentation. Der von newHttpClient () bereitgestellte HTTP-Client generiert Traces für ausgeführte Anfragen.

```
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();
  }
}
```

Instrumentierungsunterstützung für andere Bibliotheken

Die vollständige Liste der unterstützten Bibliotheksinstrumentierungen für OpenTelemetry Java finden Sie im entsprechenden GitHub Instrumentierungs-Repository unter Unterstützte Bibliotheken, Frameworks, Anwendungsserver und JVMs.

Alternativ können Sie in der OpenTelemetry Registry nachsehen, ob Instrumentierung OpenTelemetry unterstützt wird. Informationen zum Starten der Suche finden Sie unter Registrierung.

Manuelles Erstellen von Trace-Daten

With X-Ray SDK

Mit dem X-Ray-SDK sind die `beginSubsegment` Methoden `beginSegment` und erforderlich, um X-Ray-Segmente und -Untersegmente manuell zu erstellen.

```
 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

Sie können benutzerdefinierte Zeitspannen verwenden, um die Leistung interner Aktivitäten zu überwachen, die nicht von Instrumentenbibliotheken erfasst werden. Beachten Sie, dass nur Span-Kind-Server in X-Ray-Segmente konvertiert werden, alle anderen Spans werden in X-Ray-Untersegmente umgewandelt.

Zunächst müssen Sie einen Tracer erstellen, um Spans zu generieren, die Sie mit dieser Methode abrufen können. `openTelemetry.getTracer` Dadurch wird eine Tracer-Instanz aus dem

bereitgestelltTracerProvider, der im Beispiel global registriert wurde. Lösungen für die manuelle Instrumentierung mit dem SDK Sie können so viele Tracer-Instanzen wie nötig erstellen, aber es ist üblich, einen Tracer für eine gesamte Anwendung zu haben.

```
Tracer tracer = openTelemetry.getTracer("my-app");
```

Sie können den Tracer verwenden, um Spans zu erstellen.

```
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();
```

Spans haben den Standardtyp INTERNAL.

```
// Default span of type INTERNAL will become an X-Ray subsegment
Span span = tracer.spanBuilder("process-header")
  .startSpan();
try (Scope ignored = span.makeCurrent()) {
```

```
  doProcessHeader();
}
```

Hinzufügen von Anmerkungen und Metadaten zu Traces mit dem SDK OpenTelemetry

Im obigen Beispiel wird die `setAttribute` Methode verwendet, um jedem Bereich Attribute hinzuzufügen. Standardmäßig werden alle Span-Attribute in Metadaten in X-Rohdaten umgewandelt. Um sicherzustellen, dass ein Attribut in eine Anmerkung und nicht in Metadaten umgewandelt wird, fügt das obige Beispiel den Schlüssel dieses Attributs zur Liste des `aws.xray.annotations` Attributs hinzu. Weitere Informationen finden Sie unter Aktivieren der benutzerdefinierten X-Ray-Anmerkungen und Anmerkungen und Metadaten.

Mit OpenTelemetry basierten Java-Agenten

Wenn Sie den Java-Agenten zur automatischen Instrumentierung Ihrer Anwendung verwenden, müssen Sie in Ihrer Anwendung eine manuelle Instrumentierung durchführen. Zum Beispiel, um Code innerhalb der Anwendung für Abschnitte zu instrumentieren, die von keiner Bibliothek für automatische Instrumentierung abgedeckt werden.

Um eine manuelle Instrumentierung mit dem Agenten durchzuführen, müssen Sie das `opentelemetry-api` Artefakt verwenden. Die Artefaktversion darf nicht neuer als die Agentenversion sein.

```
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();
```

Lambda-Instrumentierung

With X-Ray SDK

Wenn Sie das X-Ray-SDK verwenden, ist nach der Aktivierung von Active Tracing auf Ihrem Lambda keine zusätzliche Konfiguration erforderlich, um das X-Ray-SDK zu verwenden. Lambda erstellt ein Segment, das den Lambda-Handler-Aufruf darstellt, und Sie können Untersegmente oder Instrumentenbibliotheken mit dem X-Ray SDK ohne zusätzliche Konfiguration erstellen.

With OpenTelemetry-based solutions

Lambda-Schichten mit automatischer Instrumentierung — Mithilfe der folgenden Lösungen können Sie Ihr Lambda automatisch mit AWS Lambda-Schichten von Drittanbietern instrumentieren:

- CloudWatch Lambda-Schicht für Anwendungssignale (empfohlen)

> **Note**
>
> Auf dieser Lambda-Schicht sind CloudWatch Application Signals standardmäßig aktiviert, wodurch die Leistungs- und Zustandsüberwachung für Ihre Lambda-Anwendung ermöglicht wird, indem sowohl Metriken als auch Traces erfasst werden. Um nur die Ablaufverfolgung zu ermöglichen, legen Sie die Umgebungsvariable Lambda fest. `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`

  - Ermöglicht die Leistungs- und Zustandsüberwachung für Ihre Lambda-Anwendung
  - Sammelt standardmäßig sowohl Metriken als auch Traces
- AWS verwaltete Lambda-Schicht für ADOT Java. Weitere Informationen finden Sie unter AWS Distro for OpenTelemetry Lambda Support for Java.

Informationen zur Verwendung der manuellen Instrumentierung zusammen mit der Ebene für automatische Instrumentierung finden Sie unterLösungen für die manuelle Instrumentierung mit dem SDK. Um weniger Kaltstarts zu erzielen, sollten Sie erwägen, OpenTelemetry manuelle Instrumente zu verwenden, um OpenTelemetry Traces für Ihre Lambda-Funktion zu generieren.

OpenTelemetry manuelle Instrumentierung für AWS Lambda

Stellen Sie sich den folgenden Lambda-Funktionscode vor, der einen Amazon S3 ListBuckets S3-Aufruf durchführt.

```
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);
        }
    }
}
```

Hier sind die Abhängigkeiten.

```
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')
}
```

Gehen Sie wie folgt vor, um Ihren Lambda-Handler und den Amazon S3 S3-Client manuell zu instrumentieren.

1. Ersetzen Sie Ihre Funktionsklassen, die `RequestHandler` (oder RequestStreamHandler) implementieren, durch solche, die `TracingRequestHandler` (oder TracingRequestStreamHandler) erweitern.
2. Instanziieren Sie ein Objekt TracerProvider und registrieren Sie es global. OpenTelemetrySdk TracerProvider Es wird empfohlen, das zu konfigurieren mit:
   a. Ein einfacher Span-Prozessor mit einem X-Ray-UDP-Span-Exporter zum Senden von Traces an den UDP-X-Ray-Endpunkt von Lambda
   b. Ein ParentBased Always-On-Sampler (Standard, falls nicht konfiguriert)
   c. Eine Ressource, bei der service.name auf den Lambda-Funktionsnamen gesetzt ist
   d. Ein Röntgen-Lambda-Propagator
3. Ändern Sie die `handleRequest` Methode in `doHandleRequest` und übergeben Sie das `OpenTelemetrySdk` Objekt an die Basisklasse.
4. Instrumentieren Sie den Amazon S3 S3-Client mit der OpenTemetry AWS SDK-Instrumentierung, indem Sie den Interceptor bei der Erstellung des Clients registrieren.

Sie benötigen die folgenden zugehörigen Abhängigkeiten OpenTelemetry.

```
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')
}
```

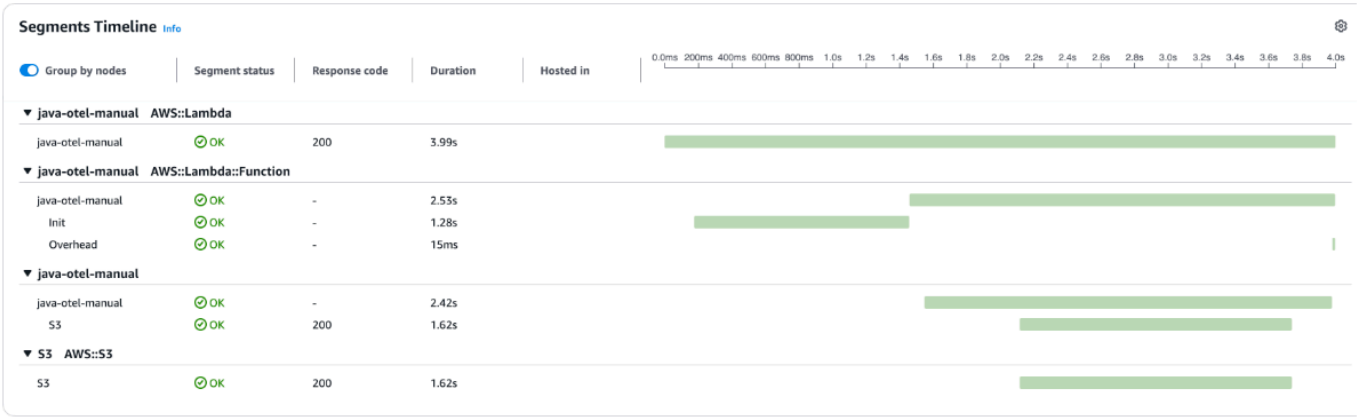
Der folgende Code demonstriert die Lambda-Funktion nach den erforderlichen Änderungen. Sie können zusätzliche benutzerdefinierte Bereiche erstellen, um die automatisch bereitgestellten Bereiche zu ergänzen.

```
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);
        }
    }
}
```

Wenn Sie die Lambda-Funktion aufrufen, sehen Sie in der Konsole unter Trace Map den CloudWatch folgenden Trace.



Zu OpenTelemetry Go migrieren

Verwenden Sie die folgenden Codebeispiele, um Ihre Go-Anwendungen bei der Migration von X-Ray manuell mit dem OpenTelemetry SDK zu instrumentieren.

Manuelle Instrumentierung mit dem SDK

Tracing setup with X-Ray SDK

Bei der Verwendung des X-Ray-SDK SDK for Go mussten Service-Plugins oder lokale Sampling-Regeln konfiguriert werden, bevor Ihr Code instrumentiert wurde.

```
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

Konfigurieren Sie das OpenTelemetry SDK, indem Sie a instanziieren TracerProvider und es als globalen Tracer-Anbieter registrieren. Wir empfehlen die Konfiguration der folgenden Komponenten:

- OTLP Trace Exporter — Erforderlich für den Export von Traces an den CloudWatch Agent oder Collector OpenTelemetry
- X-Ray Propagator — Erforderlich für die Weitergabe des Trace-Kontextes an in X-Ray integrierte AWS Dienste
- X-Ray Remote Sampler — Erforderlich für Probenentnahmeanfragen unter Verwendung von Röntgenprobenregeln
- Ressourcendetektoren — Um Metadaten des Hosts zu erkennen, auf dem Ihre Anwendung ausgeführt wird

```
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
}
```

Nachverfolgung eingehender Anfragen (HTTP-Handler-Instrumentierung)

With X-Ray SDK

Um einen HTTP-Handler mit X-Ray zu instrumentieren, wurde die X-Ray-Handler-Methode verwendet, um Segmente zu generieren mit 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

Um einen HTTP-Handler zu instrumentieren OpenTelemetry, verwenden Sie die NewHandler-Methode OpenTelemetry von, um Ihren ursprünglichen Handler-Code zu umschließen.

```
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 SDK for Go v2-Instrumentierung

With X-Ray SDK

Um ausgehende AWS Anfragen vom AWS SDK zu instrumentieren, wurden Ihre Kunden wie folgt instrumentiert:

```
// 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

Die Ablaufverfolgungsunterstützung für AWS Downstream-SDK-Aufrufe wird vom OpenTelemetry AWS SDK for Go v2 Instrumentation bereitgestellt. Hier ist ein Beispiel für die Nachverfolgung eines S3-Client-Anrufs:

```
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
    }
```

Instrumentieren von ausgehenden HTTP-Aufrufen

With X-Ray SDK

Um ausgehende HTTP-Aufrufe mit X-Ray zu instrumentieren, wurde der Xray.Client verwendet, um eine Kopie eines bereitgestellten HTTP-Clients zu erstellen.

```
myClient := xray.Client(http-client)

resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

With OpenTelemetry SDK

Um HTTP-Clients mit zu instrumentieren OpenTelemetry, verwenden Sie OpenTelemetry otelhttp. NewTransport Methode, um das HTTP zu umschließen. 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)
```

Instrumentierungsunterstützung für andere Bibliotheken

Die vollständige Liste der unterstützten Bibliotheksinstrumentierungen für OpenTelemetry Go finden Sie unter Instrumentierungspakete.

Alternativ können Sie in der OpenTelemetry Registry unter Registrierung herausfinden, ob Instrumentierung für Ihre Bibliothek OpenTelemetry unterstützt wird.

Manuelles Erstellen von Trace-Daten

With X-Ray SDK

Mit dem X-Ray-SDK waren die BeginSubsegment Methoden BeginSegment und erforderlich, um X-Ray-Segmente und Untersegmente manuell zu erstellen.

```
// 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

Verwenden Sie benutzerdefinierte Zeitspannen, um die Leistung interner Aktivitäten zu überwachen, die nicht in den Instrumentenbibliotheken erfasst werden. Beachten Sie, dass nur Bereiche der Art Server in X-Ray-Segmente umgewandelt werden, alle anderen Bereiche werden in X-Ray-Untersegmente umgewandelt.

Zunächst müssen Sie einen Tracer erstellen, um Spannen zu generieren, die Sie mit der Methode abrufen können. `otel.Tracer` Dadurch wird eine Tracer-Instanz aus dem bereitgestellt TracerProvider , der im Tracing-Setup-Beispiel global registriert wurde. Sie können so viele Tracer-Instanzen wie nötig erstellen, aber es ist üblich, einen Tracer für eine gesamte Anwendung zu verwenden.

```
    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()
```

Hinzufügen von Anmerkungen und Metadaten zu Traces mit dem SDK OpenTelemetry

Im obigen Beispiel wird die `WithAttributes` Methode verwendet, um jedem Bereich Attribute hinzuzufügen. Beachten Sie, dass standardmäßig alle Span-Attribute in X-Rohdaten in Metadaten umgewandelt werden. Um sicherzustellen, dass ein Attribut in eine Anmerkung und nicht in Metadaten umgewandelt wird, fügen Sie den Schlüssel des Attributs zur Liste des `aws.xray.annotations` Attributs hinzu. Weitere Informationen finden Sie unter Aktivieren der benutzerdefinierten X-Ray-Anmerkungen.

Manuelle Lambda-Instrumentierung

With X-Ray SDK

Mit dem X-Ray-SDK waren nach der Aktivierung von Active Tracing auf Ihrem Lambda keine zusätzlichen Konfigurationen erforderlich, um das X-Ray-SDK zu verwenden. Lambda hat ein

Segment erstellt, das den Lambda-Handler-Aufruf darstellt, und Sie haben Untersegmente mit dem X-Ray SDK ohne zusätzliche Konfiguration erstellt.

With OpenTelemetry SDK

Der folgende Lambda-Funktionscode (ohne Instrumentierung) führt einen Amazon S3 ListBuckets S3-Aufruf und eine ausgehende HTTP-Anfrage aus.

```
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)
}
```

Gehen Sie wie folgt vor, um Ihren Lambda-Handler und den Amazon S3 S3-Client manuell zu instrumentieren:

1. Instanziieren Sie in main () a TracerProvider (tp) und registrieren Sie es als globalen Tracer-Anbieter. TracerProvider Es wird empfohlen, den zu konfigurieren mit:
   a. Einfacher Span-Prozessor mit einem X-Ray-UDP-Span-Exporter zum Senden von Traces an den UDP-X-Ray-Endpunkt von Lambda
   b. Ressource, bei der service.name auf den Lambda-Funktionsnamen gesetzt ist
2. Ändern Sie die Verwendung von to. `lambda.Start(lambdaHandler)` `lambda.Start(otellambda.InstrumentHandler(lambdaHandler, xrayconfig.WithRecommendedOptions(tp)...))`
3. Instrumentieren Sie den Amazon S3 S3-Client mit der OpenTemetry AWS SDK-Instrumentierung, indem Sie OpenTelemetry Middleware für an `aws-sdk-go-v2` die Amazon S3 S3-Client-Konfiguration anhängen.
4. Instrumentieren Sie den HTTP-Client, indem Sie die `otelhttp.NewTransport` Methode verwenden, um OpenTelemetry den zu umschließen. `http.DefaultTransport`

Der folgende Code ist ein Beispiel dafür, wie die Lambda-Funktion nach den Änderungen aussehen wird. Sie können zusätzlich zu den automatisch bereitgestellten Bereichen manuell weitere benutzerdefinierte Bereiche erstellen.

```
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)...))
}
```

Wenn Sie Lambda aufrufen, sehen Sie in der Konsole den folgenden Trace: `Trace Map` CloudWatch

Migrieren Sie zu Node.js OpenTelemetry

In diesem Abschnitt wird erklärt, wie Sie Ihre Node.js -Anwendungen vom X-Ray SDK auf migrieren OpenTelemetry. Er behandelt sowohl automatische als auch manuelle Instrumentierungsansätze und bietet konkrete Beispiele für gängige Anwendungsfälle.

Das X-Ray Node.js SDK hilft Ihnen, Ihre Node.js -Anwendungen manuell für die Ablaufverfolgung zu instrumentieren. Dieser Abschnitt enthält Codebeispiele für die Migration von X-Ray zu OpenTelemetry Instrumentation.

Sections

Automatische Instrumentierungslösungen ohne Code

Um Anfragen mit dem X-Ray SDK für Node.js nachzuverfolgen, müssen Sie Ihren Anwendungscode ändern. Mit OpenTelemetry können Sie automatische Instrumentierungslösungen ohne Code verwenden, um Anfragen zu verfolgen.

Automatische Null-Code-Instrumentierung mit OpenTelemetry basierten automatischen Instrumentierungen.

1. Verwenden der automatischen Instrumentierung von AWS Distro for OpenTelemetry (ADOT) für Node.js — Informationen zur automatischen Instrumentierung für die Anwendung Node.js finden Sie unter Tracing and Metrics with the AWS Distro for Auto-Instrumentation. OpenTelemetry JavaScript

   (Optional) Sie können CloudWatch Application Signals auch bei der automatischen Instrumentierung Ihrer Anwendungen AWS mit der JavaScript automatischen ADOT-Instrumentierung aktivieren, um den aktuellen Anwendungsstatus zu überwachen und die

langfristige Anwendungsleistung anhand Ihrer Geschäftsziele zu verfolgen. Application Signals bietet Ihnen einen einheitlichen, anwendungsorientierten Überblick über Ihre Anwendungen, Services und Abhängigkeiten und unterstützt Sie bei der Überwachung und Diagnose des Zustands Ihrer Anwendungen. Weitere Informationen finden Sie unter Application Signals.

2. Verwendung der automatischen OpenTelemetry JavaScript Null-Code-Instrumentierung — Informationen zur automatischen Instrumentierung mit der finden Sie unter Zero-Code-Instrumentierung. OpenTelemetry JavaScript JavaScript

Lösungen für die manuelle Instrumentierung

Tracing setup with X-Ray SDK

Wenn das X-Ray-SDK für Node.js verwendet wurde, musste das `aws-xray-sdk` Paket das X-Ray-SDK mit Service-Plug-ins oder lokalen Sampling-Regeln konfigurieren, bevor das SDK zur Instrumentierung Ihres Codes verwendet werden konnte.

```
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 X-Ray Remote Sampling kann derzeit nicht für OpenTelemetry JS konfiguriert werden. Unterstützung für X-Ray Remote Sampling ist derzeit jedoch über die ADOT Auto-Instrumentation for Node.js verfügbar.

Für das folgende Codebeispiel benötigen Sie die folgenden Abhängigkeiten:

```
npm install --save \
  @opentelemetry/api \
  @opentelemetry/sdk-node \
  @opentelemetry/exporter-trace-otlp-proto \
```

```
  @opentelemetry/propagator-aws-xray \
  @opentelemetry/resource-detector-aws
```

Sie müssen das OpenTelemetry SDK einrichten und konfigurieren, bevor Sie Ihren Anwendungscode ausführen können. Dies kann mit dem Flag —-require geschehen. Erstellen Sie eine Datei mit dem Namen instrumentation.js, die Ihre OpenTelemetry Instrumentierungskonfiguration und Ihr Setup enthält.

Es wird empfohlen, die folgenden Komponenten zu konfigurieren:

- OTLPTraceExporter — Erforderlich für den Export von Traces zum CloudWatch OpenTelemetry Agent/Collector
- AWSXRayPropagator — Erforderlich für die Weitergabe des Trace-Kontextes an AWS Dienste, die in X-Ray integriert sind
- Ressourcendetektoren (z. B. Amazon EC2 Resource Detector) — Um Metadaten des Hosts zu erkennen, auf dem Ihre Anwendung ausgeführt wird

```
/*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();
```

Anschließend können Sie Ihre Anwendung mit Ihrem OpenTelemetry Setup wie folgt ausführen:

```
node --require ./instrumentation.js app.js
```

Sie können die Instrumentierung der OpenTelemetry SDK-Bibliothek verwenden, um automatisch Spans für Bibliotheken wie das AWS SDK zu erstellen. Wenn Sie diese aktivieren, werden automatisch Spans für Module wie das AWS SDK für Version 3 erstellt. JavaScript OpenTelemetry bietet die Möglichkeit, alle Bibliotheksinstrumentierungen zu aktivieren oder anzugeben, welche Bibliotheksinstrumentierungen aktiviert werden sollen.

Um alle Instrumentierungen zu aktivieren, installieren Sie das Paket: `@opentelemetry/auto-instrumentations-node`

```
npm install @opentelemetry/auto-instrumentations-node
```

Aktualisieren Sie als Nächstes die Konfiguration, um alle Bibliotheksinstrumentierungen zu aktivieren, wie unten gezeigt.

```
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

Sie können die automatische ADOT-Instrumentierung für Node.js verwenden, um Ihre Node.js -Anwendungen automatisch zu konfigurieren OpenTelemetry . Durch die Verwendung von ADOT Auto-Instrumentation müssen Sie keine manuellen Codeänderungen vornehmen, um eingehende Anfragen oder Bibliotheken wie AWS SDK- oder HTTP-Clients nachzuverfolgen. Weitere Informationen finden Sie unter Tracing und Metriken mit der AWS Distro for Auto-Instrumentation. OpenTelemetry JavaScript

Die automatische ADOT-Instrumentierung für Node.js unterstützt:

- Röntgenfernprobenentnahme durch Umgebungsvariable — `export OTEL_TRACES_SAMPLER=xray`
- Übertragung des X-Ray-Trace-Kontextes (standardmäßig aktiviert)
- Ressourcenerkennung (Ressourcenerkennung für Amazon- EC2, Amazon ECS- und Amazon EKS-Umgebungen ist standardmäßig aktiviert)
- Automatische Bibliotheksinstrumentierung für alle unterstützten OpenTelemetry Instrumentierungen, die disabled/enabled selektiv über `OTEL_NODE_ENABLED_INSTRUMENTATIONS` Variablen und Umgebungsvariablen gesteuert werden können `OTEL_NODE_DISABLED_INSTRUMENTATIONS`
- Manuelles Erstellen von Spans

Nachverfolgung eingehender Anfragen

With X-Ray SDK

Express.js

Mit dem X-Ray-SDK zur Verfolgung eingehender HTTP-Anfragen, die von den Anwendungen Express.js empfangen `AWSXRay.express.closeSegment()` wurden, mussten die beiden Middlewares `AWSXRay.express.openSegment(<name>)` und die beiden Middlewares alle Ihre definierten Routen umschließen, um sie zu verfolgen.

```
app.use(xrayExpress.openSegment('defaultName'));

...
```

```
app.use(xrayExpress.closeSegment());
```

Erneut stifizieren

Um eingehende HTTP-Anfragen zu verfolgen, die von `Restify` Anwendungen empfangen wurden, wurde die Middleware aus dem X-Ray SDK verwendet, indem enable über das `aws-xray-sdk-restify` Modul auf dem Restify-Server ausgeführt wurde:

```
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

Die Ablaufverfolgungsunterstützung für eingehende Anfragen für `Express.js` wird durch die OpenTelemetry HTTP-Instrumentierung und die OpenTelemetry Express-Instrumentierung bereitgestellt. Installieren Sie die folgenden Abhängigkeiten mit`npm`:

```
npm install --save @opentelemetry/instrumentation-http @opentelemetry/
instrumentation-express
```

Aktualisieren Sie die OpenTelemetry SDK-Konfiguration, um die Instrumentierung für das Express-Modul zu aktivieren:

```
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(),
  ],
});
```

Erneut stifizieren

Für Restify-Anwendungen benötigen Sie die Restify-Instrumentierung. OpenTelemetry
Installieren Sie die folgende Abhängigkeit:

```
npm install --save @opentelemetry/instrumentation-restify
```

Aktualisieren Sie die OpenTelemetry SDK-Konfiguration, um die Instrumentierung für das Restify-Modul zu aktivieren:

```
const { RestifyInstrumentation } = require('@opentelemetry/instrumentation-
restify');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new RestifyInstrumentation(),
  ],
});
```

AWS SDK V3-Instrumentierung JavaScript

With X-Ray SDK

Um ausgehende AWS Anfragen vom AWS SDK zu instrumentieren, haben Sie Clients wie im folgenden Beispiel instrumentiert:

```
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

Die Unterstützung der Ablaufverfolgung für AWS Downstream-SDK-Aufrufe an DynamoDB, Amazon S3 und andere wird durch die OpenTelemetry AWS SDK-Instrumentierung bereitgestellt. Installieren Sie die folgende Abhängigkeit mit: `npm`

```
npm install --save @opentelemetry/instrumentation-aws-sdk
```

Aktualisieren Sie die OpenTelemetry SDK-Konfiguration mit der AWS SDK-Instrumentierung.

```
import { AwsInstrumentation } from '@opentelemetry/instrumentation-aws-sdk';
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new AwsInstrumentation()
```

```
  ],
});
```

Instrumentieren von ausgehenden HTTP-Aufrufen

With X-Ray SDK

Um ausgehende HTTP-Anfragen mit X-Ray zu instrumentieren, war es erforderlich, Clients zu instrumentieren. Siehe zum Beispiel unten.

Einzelne HTTP-Clients

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Alle HTTP-Clients (global)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

With OpenTelemetry SDK

Die Ablaufverfolgungsunterstützung für die HTTP-Clients von Node.js wird von der OpenTelemetry HTTP Instrumentation bereitgestellt. Installieren Sie die folgende Abhängigkeit mit`npm`:

```
npm install --save @opentelemetry/instrumentation-http
```

Aktualisieren Sie die OpenTelemetry SDK-Konfiguration wie folgt:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
...
```

```
const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new HttpInstrumentation(),
  ],
});
```

Instrumentierungsunterstützung für andere Bibliotheken

Die vollständige Liste der unterstützten Bibliotheksinstrumentationen finden Sie OpenTelemetry JavaScript unter Unterstützte Instrumentationen.

Alternativ können Sie in der OpenTelemetry Registry unter Registrierung herausfinden, ob Instrumentierung für Ihre Bibliothek OpenTelemetry unterstützt wird.

Manuelles Erstellen von Trace-Daten

With X-Ray SDK

Bei Verwendung von X-Ray war der `aws-xray-sdk` Paketcode erforderlich, um Segmente und ihre untergeordneten Untersegmente manuell zu erstellen, um Ihre Anwendung zu verfolgen.

```
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

Sie können benutzerdefinierte Zeitspannen erstellen und verwenden, um die Leistung interner Aktivitäten zu überwachen, die nicht von Instrumentierungsbibliotheken erfasst werden. Beachten Sie, dass nur Bereiche der Art Server in X-Ray-Segmente umgewandelt werden, alle anderen Bereiche werden in X-Ray-Untersegmente umgewandelt. Weitere Informationen hierzu finden Sie unter Segmente.

Nachdem Sie das OpenTelemetry SDK im Tracing-Setup konfiguriert haben, benötigen Sie eine Tracer-Instanz, um Spans zu erstellen. Sie können so viele Tracer-Instanzen wie nötig erstellen, es ist jedoch üblich, einen Tracer für eine gesamte Anwendung zu verwenden.

```
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();
  });
```

Hinzufügen von Anmerkungen und Metadaten zu Traces mit dem SDK OpenTelemetry

Sie können Ihren Spans auch benutzerdefinierte Schlüssel-Wert-Paare als Attribute hinzufügen. Beachten Sie, dass alle diese Span-Attribute standardmäßig in Metadaten in X-Rohdaten umgewandelt werden. Um sicherzustellen, dass ein Attribut in eine Anmerkung und nicht

in Metadaten umgewandelt wird, fügen Sie den Schlüssel des Attributs zur Liste des `aws.xray.annotations` Attributs hinzu. Weitere Informationen finden Sie unter Aktivieren der benutzerdefinierten X-Ray-Anmerkungen.

```
 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();
 });
```

Lambda-Instrumentierung

With X-Ray SDK

Nachdem Sie Active Tracing für Ihre Lambda-Funktion aktiviert hatten, war das X-Ray SDK ohne zusätzliche Konfiguration erforderlich. Lambda erstellt ein Segment, das den Lambda-Handler-Aufruf darstellt, und Sie haben Untersegmente oder Instrumentenbibliotheken mit dem X-Ray SDK ohne zusätzliche Konfiguration erstellt.

With OpenTelemetry SDK

Sie können Ihr Lambda automatisch mit verkauften AWS Lambda-Layern instrumentieren. Es gibt zwei Lösungen:

- (Empfohlen) Die Lambda-Schicht wird von der CloudWatch Anwendung signalisiert

> **Note**
>
> Auf dieser Lambda-Schicht sind CloudWatch Application Signals standardmäßig aktiviert, wodurch die Leistungs- und Zustandsüberwachung für Ihre Lambda-Anwendung ermöglicht wird, indem sowohl Metriken als auch Traces erfasst werden. Wenn Sie nur die Ablaufverfolgung wünschen, sollten Sie die Lambda-

Umgebungsvariable festlegen. `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false` Weitere Informationen finden Sie unter Aktivieren Ihrer Anwendungen auf Lambda.

- AWS verwaltete Lambda-Schicht für ADOT JS. Weitere Informationen finden Sie unter AWS Distro for OpenTelemetry Lambda Support For. JavaScript

Manuelles Erstellen von Spans mit Lambda-Instrumentierung

Der ADOT JavaScript Lambda Layer bietet zwar automatische Instrumentierung für Ihre Lambda-Funktion, Sie müssen jedoch möglicherweise eine manuelle Instrumentierung in Ihrem Lambda durchführen, um beispielsweise benutzerdefinierte Daten bereitzustellen oder Code innerhalb der Lambda-Funktion selbst zu instrumentieren, der nicht durch Bibliotheksinstrumentierungen abgedeckt wird.

Um neben der automatischen Instrumentierung auch eine manuelle Instrumentierung durchzuführen, müssen Sie eine Abhängigkeit hinzufügen. `@opentelemetry/api` Es wird empfohlen, dass es sich bei der Version dieser Abhängigkeit um dieselbe Version derselben Abhängigkeit handelt, die vom ADOT JavaScript SDK verwendet wird. Sie können die OpenTelemetry API verwenden, um manuell Spans in Ihrer Lambda-Funktion zu erstellen.

Um die `@opentelemetry/api` Abhängigkeit mit NPM hinzuzufügen:

```
npm install @opentelemetry/api
```

Migrieren Sie zu .NET OpenTelemetry

Wenn Sie X-Ray Tracing in Ihren .NET-Anwendungen verwenden, wird das X-Ray-.NET-SDK mit manuellem Aufwand für die Instrumentierung verwendet.

Dieser Abschnitt enthält Codebeispiele im Lösungen für die manuelle Instrumentierung mit dem SDK Abschnitt für die Migration von der manuellen X-Ray-Instrumentierungslösung zu OpenTelemetry manuellen Instrumentierungslösungen für.NET. Alternativ können Sie von manueller X-Ray-Instrumentierung zu OpenTelemetry automatischen Instrumentierungslösungen für Instrumenten-.NET-Anwendungen migrieren, ohne den Quellcode der Anwendung in Automatische Instrumentierungslösungen ohne Code diesem Abschnitt ändern zu müssen.

Sections

Automatische Instrumentierungslösungen ohne Code

OpenTelemetry bietet automatische Instrumentierungslösungen ohne Code. Diese Lösungen verfolgen Anfragen, ohne dass Änderungen an Ihrem Anwendungscode erforderlich sind.

OpenTelemetrybasierte automatische Instrumentierungsoptionen

1. Verwenden der AWS Distro for OpenTelemetry (ADOT) Auto-Instrumentation für .NET — Informationen zur automatischen Instrumentierung von.NET-Anwendungen finden Sie unter Tracing and Metrics with the AWS Distro for .NET Auto-Instrumentation. OpenTelemetry

   (Optional) Aktivieren Sie CloudWatch Application Signals, wenn Sie Ihre Anwendungen AWS mit der automatischen ADOT.NET-Instrumentierung automatisch instrumentieren, um:

   - Überwachen Sie den aktuellen Zustand der Anwendung
   - Verfolgen Sie die langfristige Anwendungsleistung anhand der Geschäftsziele
   - Verschaffen Sie sich einen einheitlichen, anwendungsorientierten Überblick über Ihre Anwendungen, Services und Abhängigkeiten
   - Überwachen und prüfen Sie den Zustand Ihrer Anwendungen

   Weitere Informationen finden Sie unter Application Signals.
2. Verwenden der automatischen Null-Code-Instrumentierung OpenTelemetry von.Net — Informationen zur automatischen Instrumentierung OpenTelemetry mit.Net finden Sie unter Tracing and Metrics with the AWS Distro for .NET Auto-Instrumentation. OpenTelemetry

Lösungen für die manuelle Instrumentierung mit dem SDK

Tracing configuration with X-Ray SDK

Für .NET-Webanwendungen wird das X-Ray-SDK im Abschnitt AppSettings der `Web.config` Datei konfiguriert.

Beispiel Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
  </appSettings>
</configuration>
```

Für .NET Core wird eine Datei `appsettings.json` mit einem Schlüssel der obersten Ebene namens verwendet, und dann `XRay` wird ein Konfigurationsobjekt erstellt, um den X-Ray-Recorder zu initialisieren.

Beispiel für.NET `appsettings.json`

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin"
  }
}
```

Beispiel für.NET Core Program.cs — Recorder-Konfiguration

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Tracing configuration with OpenTelemetry SDK

Fügen Sie diese Abhängigkeiten hinzu:

```
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
```

Konfigurieren Sie das OpenTelemetry SDK für Ihre .NET-Anwendung, indem Sie Global einrichten TracerProvider. Die folgende Beispielkonfiguration ermöglicht auch die Instrumentierung für`ASP.NET Core`. Informationen zur `ASP.NET` Instrumentierung finden Sie unterNachverfolgung eingehender Anfragen (ASP.NET und ASP.NET-Kerninstrumentierung). Zur Verwendung OpenTelemetry mit anderen Frameworks finden Sie unter Registry weitere Bibliotheken für unterstützte Frameworks.

Es wird empfohlen, die folgenden Komponenten zu konfigurieren:

- `An OTLP Exporter`— Erforderlich für den Export von Traces zum CloudWatch Agent/ Collector OpenTelemetry
- Ein AWS X-Ray Propagator — Erforderlich für die Weitergabe des Trace-Kontextes an AWS Dienste, die in X-Ray integriert sind
- Ein AWS Röntgen-Remote-Sampler — Erforderlich, wenn Sie Anfragen anhand von Röntgenprobenahmeregeln beproben müssen
- `Resource Detectors`(zum Beispiel Amazon EC2 Resource Detector) — Um Metadaten des Hosts zu erkennen, auf dem Ihre Anwendung ausgeführt wird

```
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
```

Um sie OpenTelemetry für eine Konsolen-App zu verwenden, fügen Sie beim Start Ihres Programms die folgende OpenTelemetry Konfiguration hinzu.

```
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
```

Manuelles Erstellen von Trace-Daten

With X-Ray SDK

Mit dem X-Ray SDK waren die `BeginSubsegment` Methoden `BeginSegment` und erforderlich, um X-Ray-Segmente und Untersegmente manuell zu erstellen.

```
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

In.NET können Sie die Aktivitäts-API verwenden, um benutzerdefinierte Spans zu erstellen, um die Leistung interner Aktivitäten zu überwachen, die nicht von Instrumentierungsbibliotheken erfasst werden. Beachten Sie, dass nur Bereiche der Art Server in X-Ray-Segmente umgewandelt werden, alle anderen Bereiche werden in X-Ray-Subsegmente umgewandelt.

Sie können so viele `ActivitySource` Instanzen wie nötig erstellen, es wird jedoch empfohlen, nur eine für eine gesamte Anwendung/einen Dienst zu verwenden.

```
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
    }
}
```

Hinzufügen von Anmerkungen und Metadaten zu Traces mit dem SDK OpenTelemetry

Sie können Ihren Spans auch benutzerdefinierte Schlüssel-Wert-Paare als Attribute hinzufügen, indem Sie die `SetTag` Methode für eine Aktivität verwenden. Beachten Sie, dass standardmäßig alle Span-Attribute in X-Rohdaten in Metadaten umgewandelt werden. Um sicherzustellen, dass ein Attribut in eine Anmerkung und nicht in Metadaten umgewandelt wird, können Sie den Schlüssel dieses Attributs zur `aws.xray.annotations` Attributliste hinzufügen.

```
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
    }
}
```

Mit OpenTelemetry automatischer Instrumentierung

Wenn Sie eine OpenTelemetry automatische Instrumentierungslösung für .NET verwenden und in Ihrer Anwendung manuelle Instrumentierung durchführen müssen, um beispielsweise Code innerhalb der Anwendung selbst für Abschnitte zu instrumentieren, die nicht von einer Bibliothek für automatische Instrumentierung abgedeckt werden.

Da es nur eine globale Instrumentierung geben kann`TracerProvider`, sollte die manuelle Instrumentierung nicht ihre eigene instanziieren, `TracerProvider` wenn sie zusammen mit der automatischen Instrumentierung verwendet wird. Wenn diese `TracerProvider` Option

verwendet wird, funktioniert die benutzerdefinierte manuelle Ablaufverfolgung genauso, wenn automatische Instrumentierung oder manuelle Instrumentierung über das SDK verwendet wird. OpenTelemetry

Nachverfolgung eingehender Anfragen (ASP.NET und ASP.NET-Kerninstrumentierung)

With X-Ray SDK

Informationen zum Instrumentieren von Anfragen, die von der ASP.NET-Anwendung bedient werden, finden Sie unterhttps://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-dotnet-messagehandler.html, wie Sie die `Init` Methode Ihrer Datei aufrufen`RegisterXRay`. `global.asax`

```
AWSXRayASPNET.RegisterXRay(this, "MyApp");
```

Um Anfragen zu instrumentieren, die von Ihrer ASP.NET-Kernanwendung bedient werden, wird die `UseXRay` Methode vor jeder anderen Middleware in der `Configure` Methode Ihrer Startup-Klasse aufgerufen.

```
app.UseXRay("MyApp");
```

With OpenTelemetry SDK

OpenTelemetry stellt außerdem Instrumentierungsbibliotheken zur Erfassung von Traces für eingehende Webanfragen für ASP.NET und ASP.NET Core bereit. Im folgenden Abschnitt werden die Schritte aufgeführt, die erforderlich sind, um diese Bibliotheksinstrumentierungen für Ihre OpenTelemetry Konfiguration hinzuzufügen und zu aktivieren. Dazu gehört auch, wie Sie beim Erstellen des Tracer-Providers ASP.NET - oder ASP.NET-Kerninstrumentierung hinzufügen.

Für Informationen zur Aktivierung von .Instrumentation. OpenTelemetry AspNet, siehe Schritte zur Aktivierung von OpenTelemetry .Instrumentation. AspNetund für Informationen zur Aktivierung von OpenTelemetry .Instrumentation. AspNetCore, siehe Schritte zur Aktivierung von OpenTelemetry .Instrumentation. AspNetCore.

AWS SDK-Instrumentierung

With X-Ray SDK

Installieren Sie alle AWS SDK-Clients, indem Sie anrufen`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
```

Verwenden Sie eine der folgenden Methoden für eine spezifische AWS Service-Client-Instrumentierung.

```
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

Für das folgende Codebeispiel benötigen Sie die folgende Abhängigkeit:

```
dotnet add package OpenTelemetry.Instrumentation.AWS
```

Um das AWS SDK zu instrumentieren, aktualisieren Sie die OpenTelemetry SDK-Konfiguration, in der Global eingerichtet TracerProvider ist.

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
```

```
        .AddAWSInstrumentation()
        ...
```

Instrumentieren von ausgehenden HTTP-Aufrufen

With X-Ray SDK

Das X-Ray.NET-SDK verfolgt ausgehende HTTP-Aufrufe über die Erweiterungsmethoden `GetResponseTraced()` oder `GetAsyncResponseTraced()` bei der Verwendung `System.Net.HttpWebRequest` oder mithilfe des `HttpClientXRayTracingHandler` Handlers bei der Verwendung`System.Net.Http.HttpClient`.

With OpenTelemetry SDK

Für das folgende Codebeispiel benötigen Sie die folgende Abhängigkeit:

```
dotnet add package OpenTelemetry.Instrumentation.Http
```

Um die OpenTelemetry SDK-Konfiguration zu instrumentieren `System.Net.Http.HttpClient` und `System.Net.HttpWebRequest` zu aktualisieren, in der Global eingerichtet TracerProvider ist.

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
        .AddHttpClientInstrumentation()
        ...
```

Instrumentierungsunterstützung für andere Bibliotheken

Sie können die OpenTelemetry Registry nach .NET-Instrumentierungsbibliotheken durchsuchen und filtern, um herauszufinden, ob OpenTelemetry die Instrumentierung für Ihre Bibliothek unterstützt wird. Suchen Sie in der Registry nach, um mit der Suche zu beginnen.

Lambda-Instrumentierung

With X-Ray SDK

Das folgende Verfahren war erforderlich, um das X-Ray SDK mit Lambda zu verwenden:

1. Aktivieren Sie Active Tracing für Ihre Lambda-Funktion
2. Der Lambda-Dienst erstellt ein Segment, das den Aufruf Ihres Handlers darstellt
3. Erstellen Sie Untersegmente oder Instrumentenbibliotheken mit dem X-Ray SDK

With OpenTelemetry-based solutions

Sie können Ihr Lambda automatisch mit verkauften AWS Lambda-Layern instrumentieren. Es gibt zwei Lösungen:

- (Empfohlen) Die Lambda-Schicht wird von der CloudWatch Anwendung signalisiert
- Für eine bessere Leistung sollten Sie die Verwendung `OpenTelemetry Manual Instrumentation` zur Generierung von OpenTelemetry Traces für Ihre Lambda-Funktion in Betracht ziehen.

OpenTelemetry manuelle Instrumentierung für AWS Lambda

Im Folgenden finden Sie ein Beispiel für einen Lambda-Funktionscode (ohne Instrumentierung).

```
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);
        }
    }
}
```

Gehen Sie wie folgt vor, um Ihren Lambda-Handler und den Amazon S3 S3-Client manuell zu instrumentieren.

1. Instanziieren Sie a TracerProvider — TracerProvider Es wird empfohlen, das mit einem`XrayUdpSpanExporter`, einem ParentBased Always On Sampler und a `Resource` zu konfigurieren, das auf den Namen der `service.name` Lambda-Funktion gesetzt ist.
2. Instrumentieren Sie den Amazon S3 S3-Client mit der OpenTemetry AWS SDK-Instrumentierung, indem Sie aufrufen`AddAWSInstrumentation()`, um die AWS SDK-Client-Instrumentierung hinzuzufügen `TracerProvider`
3. Erstellen Sie eine Wrapper-Funktion mit derselben Signatur wie die ursprüngliche Lambda-Funktion. Rufen Sie `AWSLambdaWrapper.Trace()` API auf und übergeben `TracerProvider` Sie die ursprüngliche Lambda-Funktion und ihre Eingaben als Parameter. Stellen Sie die Wrapper-Funktion als Lambda-Handler-Eingabe ein.

Für das folgende Codebeispiel benötigen Sie die folgenden Abhängigkeiten:

```
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
```

Der folgende Code demonstriert die Lambda-Funktion nach den erforderlichen Änderungen. Sie können zusätzliche benutzerdefinierte Bereiche erstellen, um die automatisch bereitgestellten Bereiche zu ergänzen.

```
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);
        }
    }
}
```

Wenn Sie dieses Lambda aufrufen, sehen Sie den folgenden Trace in der Trace Map in der CloudWatch Konsole:

Zu OpenTelemetry Python migrieren

Dieses Handbuch hilft Ihnen bei der Migration von Python-Anwendungen vom X-Ray SDK zur OpenTelemetry Instrumentierung. Es behandelt sowohl automatische als auch manuelle Instrumentierungsansätze mit Codebeispielen für gängige Szenarien.

Sections

- Instrumentierung ausgehender HTTP-Aufrufe über Anfragen
- Instrumentierungsunterstützung für andere Bibliotheken
- Manuelles Erstellen von Trace-Daten
- Lambda-Instrumentierung

Automatische Instrumentierungslösungen ohne Code

Mit X-Ray SDK mussten Sie Ihren Anwendungscode ändern, um Anfragen zu verfolgen. OpenTelemetry bietet automatische Instrumentierungslösungen ohne Code zur Rückverfolgung von Anfragen. Mit haben Sie die Möglichkeit OpenTelemetry, automatische Instrumentierungslösungen ohne Code zu verwenden, um Anfragen zu verfolgen.

Nullcode mit basierten automatischen Instrumentierungen OpenTelemetry

1. Verwenden der automatischen Instrumentierung von AWS Distro for OpenTelemetry (ADOT) für Python — Informationen zur automatischen Instrumentierung für Python-Anwendungen finden Sie unter Tracing and Metrics with the AWS Distro for Python Auto-Instrumentation. OpenTelemetry

   (Optional) Sie können CloudWatch Application Signals auch bei der automatischen Instrumentierung Ihrer Anwendungen AWS mit der automatischen ADOT-Python-Instrumentierung aktivieren, um den aktuellen Anwendungsstatus zu überwachen und die langfristige Anwendungsleistung anhand Ihrer Geschäftsziele zu verfolgen. Application Signals bietet Ihnen einen einheitlichen, anwendungsorientierten Überblick über Ihre Anwendungen, Services und Abhängigkeiten und unterstützt Sie bei der Überwachung und Diagnose des Zustands Ihrer Anwendungen.
2. Verwendung der automatischen OpenTelemetry Python-Zero-Code-Instrumentierung — Informationen zur automatischen Instrumentierung mit OpenTelemetry Python finden Sie unter Python-Zero-Code-Instrumentierung.

Instrumentieren Sie Ihre Anwendungen manuell

Sie können Ihre Anwendungen mithilfe des `pip` Befehls manuell instrumentieren.

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
```

Initialisierung des Tracing-Setups

With X-Ray SDK

In X-Ray `xray_recorder` wird das Global initialisiert und verwendet, um Segmente und Untersegmente zu generieren.

With OpenTelemetry SDK

> **Note**
>
> X-Ray Remote Sampling kann derzeit nicht für OpenTelemetry Python konfiguriert werden. Unterstützung für X-Ray Remote Sampling ist derzeit jedoch über die ADOT Auto-Instrumentation for Python verfügbar.

In OpenTelemetry müssen Sie eine globale initialisieren. `TracerProvider` Auf diese Weise können Sie einen Tracer erwerben`TracerProvider`, mit dem Sie Spans an einer beliebigen Stelle in Ihrer Anwendung generieren können. Es wird empfohlen, die folgenden Komponenten zu konfigurieren:

- `OTLPSpanExporter`— Erforderlich für den Export von Traces zum CloudWatch Agent/ Collector OpenTelemetry
- Ein AWS X-Ray Propagator — Erforderlich für die Weitergabe des Trace-Kontextes an AWS Dienste, die in X-Ray integriert sind

```
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
```

Mit ADOT Auto-Instrumentation für Python

Sie können die automatische ADOT-Instrumentierung für Python verwenden, um Ihre Python-Anwendungen automatisch zu konfigurieren OpenTelemetry . Durch die automatische ADOT-Instrumentierung müssen Sie keine manuellen Codeänderungen vornehmen, um eingehende Anfragen zu verfolgen oder Bibliotheken wie das AWS SDK oder die HTTP-Clients zu verfolgen. Weitere Informationen finden Sie unter Tracing and Metrics with the AWS Distro for OpenTelemetry Python Auto-Instrumentation.

Die automatische ADOT-Instrumentierung für Python unterstützt:

- Röntgenfernabtastung durch die Umgebungsvariable `export OTEL_TRACES_SAMPLER=xray`

- Übertragung des X-Ray-Trace-Kontextes (standardmäßig aktiviert)
- Ressourcenerkennung (Ressourcenerkennung für Amazon- EC2, Amazon ECS- und Amazon EKS-Umgebungen ist standardmäßig aktiviert)
- Automatische Bibliotheksinstrumentierungen für alle unterstützten OpenTelemetry Instrumentierungen sind standardmäßig aktiviert. Sie können sie selektiv über die `OTEL_PYTHON_DISABLED_INSTRUMENTATIONS` Umgebungsvariable deaktivieren. (alle sind standardmäßig aktiviert)
- Manuelles Erstellen von Spans

Von X-Ray-Service-Plug-ins bis hin zu OpenTelemetry AWS Ressourcenanbietern

Das X-Ray-SDK bietet Plug-ins, die Sie hinzufügen können`xray_recorder`, um plattformspezifische Informationen aus dem gehosteten Service wie Amazon EC2, Amazon ECS und Elastic Beanstalk zu erfassen. Es ist den Resource Providers insofern ähnlich OpenTelemetry , als es die Informationen als Ressourcenattribute erfasst. Es sind mehrere Ressourcenanbieter für verschiedene AWS Plattformen verfügbar.

- Beginnen Sie mit der Installation des AWS Erweiterungspakets, `pip install opentelemetry-sdk-extension-aws`
- Konfigurieren Sie den gewünschten Ressourcendetektor. Das folgende Beispiel zeigt, wie der EC2 Amazon-Ressourcenanbieter im OpenTelemetry SDK konfiguriert wird

```
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)
```

Nachverfolgung eingehender Anfragen

With X-Ray SDK

Das X-Ray-Python-SDK unterstützt Anwendungsframeworks wie Django, Flask und Bottle bei der Verfolgung der eingehenden Anfragen für Python-Anwendungen, die auf ihnen ausgeführt werden. Dies erfolgt durch Hinzufügen `XRayMiddleware` zur Anwendung für jedes Framework.

With OpenTelemetry SDK

OpenTelemetry bietet Instrumentierungen für Django und Flask über die spezifischen Instrumentierungsbibliotheken. In ist keine Instrumentierung für Bottle verfügbar OpenTelemetry, Anwendungen können dennoch mithilfe der WSGI-Instrumentation verfolgt werden. OpenTelemetry

Für das folgende Codebeispiel benötigen Sie die folgende Abhängigkeit:

```
pip install opentelemetry-instrumentation-flask
```

Sie müssen das SDK initialisieren und das globale OpenTelemetry SDK registrieren, TracerProvider bevor Sie Instrumentierungen für Ihr Anwendungsframework hinzufügen können. Ohne sie werden die Trace-Operationen funktionieren. `no-ops` Sobald Sie den Global konfiguriert haben`TracerProvider`, können Sie den Instrumentor für Ihr Anwendungsframework verwenden. Das folgende Beispiel demonstriert eine Flask-Anwendung.

```
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 SDK-Instrumentierung

With X-Ray SDK

Das X-Ray-Python-SDK verfolgt die AWS SDK-Client-Anfrage, indem es die `botocore` Bibliothek patcht. Weitere Informationen finden Sie unter Nachverfolgen von AWS SDK-Aufrufen mit dem X-Ray-SDK für Python. In Ihrer Anwendung wird die `patch_all()` Methode verwendet, um alle Bibliotheken zu instrumentieren oder selektiv mithilfe von `boto3` Bibliotheken zu patchen. `botocore` `patch(['botocore'])` Jede der ausgewählten Methoden instrumentiert alle Boto3-Clients in Ihrer Anwendung und generiert ein Untersegment für jeden Aufruf, der mit diesen Clients getätigt wird.

With OpenTelemetry SDK

Für das folgende Codebeispiel benötigen Sie die folgende Abhängigkeit:

```
pip install opentelemetry-instrumentation-botocore
```

Verwenden Sie die OpenTelemetry Botocore-Instrumentation programmgesteuert, um alle Boto3-Clients in Ihrer Anwendung zu instrumentieren. Das folgende Beispiel demonstriert die Instrumentierung. `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}")
```

Instrumentierung ausgehender HTTP-Aufrufe über Anfragen

With X-Ray SDK

Das X-Ray-Python-SDK verfolgt ausgehende HTTP-Aufrufe anhand von Anfragen, indem es die Anforderungsbibliothek patcht. Weitere Informationen finden Sie unter Verfolgen von Aufrufen von Downstream-HTTP-Webdiensten mithilfe des X-Ray SDK für Python. In Ihrer Anwendung können Sie die `patch_all()` Methode verwenden, um alle Bibliotheken zu instrumentieren, oder indem Sie die Anforderungsbibliotheken selektiv patchen, indem Sie `patch(['requests'])` Jede der Optionen instrumentiert die `requests` Bibliothek und generiert ein Untersegment für jeden Aufruf, der über sie erfolgt. `requests`

With OpenTelemetry SDK

Für das folgende Codebeispiel benötigen Sie die folgende Abhängigkeit:

```
pip install opentelemetry-instrumentation-requests
```

Verwenden Sie die OpenTelemetry Requests Instrumentation programmgesteuert, um die Anforderungsbibliothek so zu instrumentieren, dass sie Traces für HTTP-Anfragen generiert, die von ihr in Ihrer Anwendung gestellt werden. Weitere Informationen finden Sie unter Instrumentation von OpenTelemetry Anfragen. Das folgende Beispiel zeigt die Instrumentierung der `requests` Bibliothek.

```
from opentelemetry.instrumentation.requests import RequestsInstrumentor

# Instrument Requests
RequestsInstrumentor().instrument()

...
```

```
    example_session = requests.Session()
    example_session.get(url="https://example.com")
```

Alternativ können Sie die zugrunde liegende `urllib3` Bibliothek auch instrumentieren, um HTTP-Anfragen zu verfolgen:

```
# 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")
```

Instrumentierungsunterstützung für andere Bibliotheken

Die vollständige Liste der unterstützten Bibliotheksinstrumentationen für OpenTelemetry Python finden Sie unter Unterstützte Bibliotheken, Frameworks, Anwendungsserver und JVMs.

Alternativ können Sie in der OpenTelemetry Registry suchen, um herauszufinden, ob Instrumentierung OpenTelemetry unterstützt wird. Suchen Sie in der Registry nach, um mit der Suche zu beginnen.

Manuelles Erstellen von Trace-Daten

Sie können Segmente und Untersegmente mithilfe von `xray_recorder` in Ihrer Python-Anwendung erstellen. Weitere Informationen finden Sie unter Manuelles Instrumentieren von Python-Code. Sie können den Trace-Daten auch manuell Anmerkungen und Metadaten hinzufügen.

Spans mit SDK erstellen OpenTelemetry

Verwenden Sie die `start_as_current_span` API, um einen Bereich zu starten und ihn für die Erstellung von Spans festzulegen. Beispiele zum Erstellen von Spans finden Sie unter Spans

erstellen. Sobald ein Bereich gestartet wurde und sich im aktuellen Bereich befindet, können Sie ihm weitere Informationen hinzufügen, indem Sie Attribute, Ereignisse, Ausnahmen, Links usw. hinzufügen. So wie wir in X-Ray Segmente und Untersegmente haben, gibt es auch hier verschiedene Arten von Spannen. OpenTelemetry Nur die `SERVER` Typbereiche werden in Röntgensegmente umgewandelt, während andere in Röntgenuntersegmente umgewandelt werden.

```
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
```

Hinzufügen von Anmerkungen und Metadaten zu Traces mit SDK OpenTelemetry

Das X-Ray-Python-SDK bietet separate APIs `put_metadata` Funktionen zum Hinzufügen von Anmerkungen und Metadaten zu einem Trace. `put_annotation` Im OpenTelemetry SDK sind die Anmerkungen und Metadaten einfach Attribute in einem Bereich, die über die `set_attribute` API hinzugefügt werden.

Span-Attribute, bei denen es sich um Anmerkungen zu einem Trace handeln soll, werden unter dem reservierten Schlüssel hinzugefügt, `aws.xray.annotations` dessen Wert eine Liste von Schlüssel-Wert-Annotationspaaren ist. Alle anderen Span-Attribute werden zu Metadaten für das konvertierte Segment oder Untersegment.

Wenn Sie den ADOT-Collector verwenden, können Sie außerdem konfigurieren, welche Span-Attribute in X-Ray-Anmerkungen konvertiert werden sollen, indem Sie das `indexed_attributes` in der Collector-Konfiguration angeben.

Das folgende Beispiel zeigt, wie Sie mithilfe des SDK Anmerkungen und Metadaten zu einer Ablaufverfolgung hinzufügen. OpenTelemetry

```
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")
```

Lambda-Instrumentierung

Um Ihre Lambda-Funktionen auf X-Ray zu überwachen, aktivieren Sie X-Ray und fügen der Funktionsaufrufrolle entsprechende Berechtigungen hinzu. Wenn Sie Downstream-Anfragen von Ihrer Funktion verfolgen, würden Sie den Code außerdem mit dem X-Ray-Python-SDK instrumentieren.

OpenTelemetry Für X-Ray wird empfohlen, die CloudWatch Application Signals Lambda-Schicht bei ausgeschalteten Application Signals zu verwenden. Dadurch wird Ihre Funktion automatisch instrumentiert und Spans für den Funktionsaufruf und alle nachgelagerten Anfragen Ihrer Funktion generiert. Wenn Sie neben der Ablaufverfolgung auch Anwendungssignale verwenden möchten, um den Zustand Ihrer Funktion zu überwachen, finden Sie weitere Informationen unter Aktivieren Ihrer Anwendungen auf Lambda.

- Suchen Sie den erforderlichen Lambda-Layer-ARN für Ihre Funktion aus AWS Lambda Layer for OpenTelemetry ARNs und fügen Sie ihn hinzu.
- Stellen Sie die folgenden Umgebungsvariablen für Ihre Funktion ein.

- `AWS_LAMBDA_EXEC_WRAPPER=/opt/otel-instrument`— Dadurch wird die automatische Instrumentierung für die Funktion geladen
- `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`— Dadurch wird die Überwachung von Anwendungssignalen deaktiviert

Manuelles Erstellen von Spans mit Lambda-Instrumentierung

Darüber hinaus können Sie innerhalb Ihrer Funktion benutzerdefinierte Zeitspannen generieren, um die Arbeit zu verfolgen. Sie können dies tun, indem Sie nur das `opentelemetry-api` Paket in Verbindung mit der automatischen Lambda-Layer-Instrumentierung von Application Signals verwenden.

1. Nehmen Sie das `opentelemetry-api` als Abhängigkeit in Ihre Funktion auf
2. Der folgende Codeausschnitt ist ein Beispiel für die Generierung benutzerdefinierter Spans

```
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
    }
```

Migrieren Sie zu Ruby OpenTelemetry

Verwenden Sie die folgenden Codebeispiele und Anleitungen für die manuelle OpenTelemetry Instrumentierung, um Ihre Ruby-Anwendungen vom X-Ray SDK zur Instrumentierung zu migrieren.

Sections

Instrumentieren Sie Ihre Lösungen manuell mit dem SDK

Tracing setup with X-Ray SDK

Das X-Ray-SDK für Ruby erforderte, dass Sie Ihren Code mit Service-Plug-ins konfigurieren.

```
require 'aws-xray-sdk'

XRay.recorder.configure(plugins: [:ec2, :elastic_beanstalk])
```

Tracing setup with OpenTelemetry SDK

> **Note**
>
> X-Ray Remote Sampling kann derzeit nicht für OpenTelemetry Ruby konfiguriert werden.

Platzieren Sie für eine Ruby on Rails-Anwendung Ihren Konfigurationscode in einem Rails-Initialisierer. Weitere Informationen finden Sie unter Erste Schritte mit . Für alle manuell instrumentierten Ruby-Programme müssen Sie die `OpenTelemetry::SDK.configure` Methode verwenden, um das Ruby-SDK zu konfigurieren. OpenTelemetry

Installieren Sie zunächst die folgenden Pakete:

```
bundle add opentelemetry-sdk opentelemetry-exporter-otlp opentelemetry-propagator-
xray
```

Als Nächstes konfigurieren Sie das OpenTelemetry SDK mithilfe des Konfigurationscodes, der bei der Initialisierung Ihres Programms ausgeführt wird. Es wird empfohlen, die folgenden Komponenten zu konfigurieren:

- `OTLP Exporter`— Erforderlich für den Export von Traces an den CloudWatch Agent und den OpenTelemetry Collector
- `An AWS X-Ray Propagator`— Erforderlich für die Weitergabe des Trace-Kontextes an AWS Dienste, die in X-Ray integriert sind

```
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 haben auch das Konzept der Bibliotheksinstrumentierung. Wenn Sie diese aktivieren, werden automatisch Bereiche für Bibliotheken wie das AWS SDK erstellt. OpenTelemetry bietet die Möglichkeit, alle Bibliotheksinstrumentierungen zu aktivieren oder anzugeben, welche Bibliotheksinstrumentierungen aktiviert werden sollen.

Um alle Instrumentierungen zu aktivieren, installieren Sie zuerst das Paket: `opentelemetry-instrumentation-all`

```
bundle add opentelemetry-instrumentation-all
```

Aktualisieren Sie als Nächstes die Konfiguration, um alle Bibliotheksinstrumentierungen wie unten gezeigt zu aktivieren:

```
require 'opentelemetry/instrumentation/all'
...

OpenTelemetry::SDK.configure do |c|
   ...

  c.use_all() # Enable all instrumentations
end
```

OpenTelemetry SDKs haben auch das Konzept der Bibliotheksinstrumentierung. Wenn Sie diese aktivieren, werden automatisch Bereiche für Bibliotheken wie das AWS SDK erstellt. OpenTelemetry bietet die Möglichkeit, alle Bibliotheksinstrumentierungen zu aktivieren oder anzugeben, welche Bibliotheksinstrumentierungen aktiviert werden sollen.

Um alle Instrumentierungen zu aktivieren, installieren Sie zuerst das Paket: `opentelemetry-instrumentation-all`

```
bundle add opentelemetry-instrumentation-all
```

Aktualisieren Sie als Nächstes die Konfiguration, um alle Bibliotheksinstrumentierungen wie unten gezeigt zu aktivieren:

```
require 'opentelemetry/instrumentation/all'
...
```

```
OpenTelemetry::SDK.configure do |c|
   ...

  c.use_all() # Enable all instrumentations
end
```

Nachverfolgung eingehender Anfragen (Rails-Instrumentierung)

With X-Ray SDK

Mit dem X-Ray SDK wird das X-Ray-Tracing bei der Initialisierung für das Rails-Framework konfiguriert.

Beispiel — _xray.rb config/initializers/aws

```
Rails.application.config.xray = {
  name: 'my app',
  patch: %I[net_http aws_sdk],
  active_record: true
}
```

With OpenTelemetry SDK

Installieren Sie zunächst die folgenden Pakete:

```
bundle add opentelemetry-instrumentation-rack opentelemetry-instrumentation-
rails opentelemetry-instrumentation-action_pack opentelemetry-instrumentation-
active_record opentelemetry-instrumentation-action_view
```

Aktualisieren Sie als Nächstes die Konfiguration, um die Instrumentierung für Ihre Rails-Anwendung zu aktivieren, wie unten gezeigt:

```
# 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 SDK-Instrumentierung

With X-Ray SDK

Um ausgehende AWS Anfragen vom AWS SDK zu instrumentieren, werden die AWS SDK-Clients wie im folgenden Beispiel mit X-Ray gepatcht:

```
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 Das SDK for Ruby V3 bietet Unterstützung für das Aufzeichnen und Ausgeben OpenTelemetry von Spuren. Informationen zur Konfiguration OpenTelemetry für einen Service-Client finden Sie unter Konfiguration von Observability-Funktionen im AWS SDK for Ruby.

Instrumentieren von ausgehenden HTTP-Aufrufen

Wenn Sie HTTP-Aufrufe an externe Dienste tätigen, müssen Sie die Aufrufe möglicherweise manuell instrumentieren, wenn die automatische Instrumentierung nicht verfügbar ist oder nicht genügend Details liefert.

With X-Ray SDK

Um Downstream-Aufrufe zu instrumentieren, wurde das X-Ray-SDK SDK for Ruby verwendet, um die `net/http` Bibliothek zu patchen, die Ihre Anwendung verwendet:

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http]
}

XRay.recorder.configure(config)
```

With OpenTelemetry SDK

Um die `net/http` Instrumentierung zu aktivieren OpenTelemetry, installieren Sie zunächst das `opentelemetry-instrumentation-net_http` Paket:

```
bundle add opentelemetry-instrumentation-net_http
```

Aktualisieren Sie als Nächstes die Konfiguration, um die `net/http` Instrumentierung wie folgt zu aktivieren:

```
OpenTelemetry::SDK.configure do |c|
   ...

  c.use 'OpenTelemetry::Instrumentation::Net::HTTP'
  ...

end
```

Instrumentierungsunterstützung für andere Bibliotheken

Die vollständige Liste der unterstützten Bibliotheksinstrumentationen für OpenTelemetry Ruby finden Sie unter opentelemetry-ruby-contrib.

Alternativ können Sie in der OpenTelemetry Registry suchen, um herauszufinden, ob Instrumentierung OpenTelemetry unterstützt wird. Weitere Informationen finden Sie unter Registrierung.

Manuelles Erstellen von Trace-Daten

With X-Ray SDK

Mit X-Ray mussten Sie für das `aws-xray-sdk` Paket manuell Segmente und deren untergeordnete Untersegmente erstellen, um Ihre Anwendung zu verfolgen. Möglicherweise haben Sie Ihren Segmenten oder Untersegmenten auch X-Ray-Anmerkungen und Metadaten hinzugefügt:

```
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

Sie können benutzerdefinierte Zeitspannen verwenden, um die Leistung interner Aktivitäten zu überwachen, die nicht von Instrumentenbibliotheken erfasst werden. Beachten Sie, dass nur Bereiche der Art Server in X-Ray-Segmente umgewandelt werden, alle anderen Bereiche werden in X-Ray-Untersegmente umgewandelt. Standardmäßig sind Spans. `INTERNAL`

Erstellen Sie zunächst einen Tracer, um Spans zu generieren, die Sie mit der Methode abrufen können. `OpenTelemetry.tracer_provider.tracer('<YOUR_TRACER_NAME>')` Dadurch wird eine Tracer-Instanz bereitgestellt, die global in der Konfiguration Ihrer Anwendung registriert ist. OpenTelemetry Es ist üblich, einen einzigen Tracer für eine gesamte Anwendung zu haben. Erstellen Sie einen OpenTelemetry Tracer und verwenden Sie ihn, um Spans zu erstellen:

```
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
```

Hinzufügen von Anmerkungen und Metadaten zu Traces mit SDK OpenTelemetry

Verwenden Sie die `set_attribute` Methode, um jedem Bereich Attribute hinzuzufügen. Beachten Sie, dass alle diese Span-Attribute standardmäßig in Metadaten in X-Rohdaten umgewandelt werden. Um sicherzustellen, dass ein Attribut in eine Anmerkung und nicht in Metadaten umgewandelt wird, können Sie diesen Attributschlüssel zur `aws.xray.annotations` Attributliste hinzufügen. Weitere Informationen finden Sie unter Aktivieren der benutzerdefinierten X-Ray-Anmerkungen.

```
# 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
```

Manuelle Lambda-Instrumentierung

With X-Ray SDK

Nachdem Active Tracing auf Lambda aktiviert wurde, sind keine zusätzlichen Konfigurationen erforderlich, um das X-Ray SDK zu verwenden. Lambda erstellt ein Segment, das den Lambda-Handler-Aufruf darstellt, und Sie können Untersegmente oder Instrumentenbibliotheken mit dem X-Ray SDK ohne zusätzliche Konfiguration erstellen.

With OpenTelemetry SDK

Betrachten Sie das folgende Beispiel für einen Lambda-Funktionscode (ohne Instrumentierung):

```
require 'json'
def lambda_handler(event:, context:)
    # TODO implement
    { statusCode: 200, body: JSON.generate('Hello from Lambda!') }
end
```

Um Ihr Lambda manuell zu instrumentieren, müssen Sie:

1. Füge die folgenden Edelsteine für dein Lambda hinzu

   ```
   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. Initialisieren Sie OpenTelemetry das SDK außerhalb Ihres Lambda-Handlers. Es wird empfohlen, das OpenTelemetry SDK zu konfigurieren mit:

   1. Ein einfacher Span-Prozessor mit einem X-Ray-UDP-Span-Exporter zum Senden von Traces an den UDP-X-Ray-Endpunkt von Lambda
   2. Ein Röntgen-Lambda-Propagator
   3. `service_name`Konfiguration, die auf den Namen der Lambda-Funktion gesetzt werden soll

3. Fügen Sie in Ihrer Lambda-Handler-Klasse die folgenden Zeilen hinzu, um Ihren Lambda-Handler zu instrumentieren:

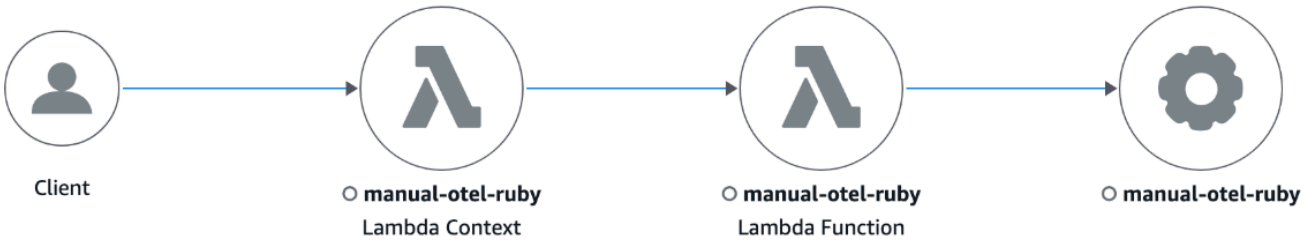
   ```
   class Handler
         extend OpenTelemetry::Instrumentation::AwsLambda::Wrap
         ...

         instrument_handler :process
   ```

```
    end
```

Der folgende Code demonstriert die Lambda-Funktion nach den erforderlichen Änderungen. Sie können zusätzliche benutzerdefinierte Bereiche erstellen, um die automatisch bereitgestellten Bereiche zu ergänzen.

```
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
```

Im Folgenden finden Sie ein Beispiel für eine Trace-Map einer instrumentierten Lambda-Funktion, die in Ruby geschrieben wurde.



Sie können Lambda-Ebenen auch verwenden, um Ihr Lambda OpenTelemetry zu konfigurieren. Weitere Informationen finden Sie unter OpenTelemetry AWS-Lambda Instrumentation.

Erstellen von X-Ray-Ressourcen mit AWS CloudFormation

AWS X-Ray ist in einen Service integriert AWS CloudFormation, der Ihnen hilft, Ihre AWS Ressourcen zu modellieren und einzurichten, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle AWS benötigten Ressourcen beschreibt und diese Ressourcen für Sie CloudFormation bereitstellt und konfiguriert.

Wenn Sie es verwenden CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre X-Ray-Ressourcen konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Ressourcen einmal und stellen Sie dann dieselben Ressourcen immer wieder in mehreren AWS-Konten Regionen bereit.

X-Ray und CloudFormation Vorlagen

Um Ressourcen für X-Ray und verwandte Dienste bereitzustellen und zu konfigurieren, müssen Sie CloudFormation Vorlagen verstehen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren CloudFormation Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie CloudFormation Designer verwenden, um Ihnen die ersten Schritte mit Vorlagen zu erleichtern. CloudFormation Weitere Informationen finden Sie unter Was ist CloudFormation -Designer? im AWS CloudFormation - Benutzerhandbuch.

X-Ray unterstützt das Erstellen von `AWS::XRay::Group`, `AWS::XRay::SamplingRule`, und `AWS::XRay::ResourcePolicy`Ressourcen in CloudFormation. Weitere Informationen, einschließlich Beispielen für JSON- und YAML-Vorlagen, finden Sie in der Referenz zum X-Ray-Ressourcentyp im AWS CloudFormation Benutzerhandbuch.

Erfahren Sie mehr über CloudFormation

Weitere Informationen CloudFormation dazu finden Sie in den folgenden Ressourcen:

- AWS CloudFormation
- AWS CloudFormation Benutzerhandbuch
- CloudFormation API Reference
- AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle

Kennzeichnen von Regeln und Gruppen für die Röntgenprobenahme

Stichwörter sind Wörter oder Ausdrücke, mit denen Sie Ihre AWS Ressourcen identifizieren und organisieren können. Sie können jeder Ressource mehrere Tags hinzufügen. Jedes Tag enthält einen Schlüssel und einen optionalen Wert, den Sie definieren. Ein Tag-Schlüssel könnte beispielsweise sein**`domain`**, und der Tag-Wert könnte sein**`example.com`**. Sie können Ihre Ressourcen anhand der von Ihnen hinzugefügten Tags suchen und filtern. Weitere Informationen zur Verwendung von Tags finden Sie unter AWS Ressourcen taggen in der AWS allgemeinen Referenz.

Sie können Tags verwenden, um tagbasierte Berechtigungen für Distributionen durchzusetzen. CloudFront Weitere Informationen finden Sie unter Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags.

> **Note**
>
> Tag Editor und AWS Resource Groups unterstützen derzeit keine X-Ray-Ressourcen. Sie können Tags mithilfe der AWS X-Ray Konsole oder der API hinzufügen und verwalten.

Sie können Tags auf Ressourcen anwenden, indem Sie die X-Ray-Konsole, API, AWS CLI SDKs, und verwenden AWS Tools for Windows PowerShell. Weitere Informationen finden Sie in der folgenden -Dokumentation:

- X-Ray-API — Die folgenden Operationen finden Sie in der AWS X-Ray API-Referenz:
  - ListTagsForResource
  - CreateSamplingRule
  - CreateGroup
  - TagResource
  - UntagResource
- AWS CLI — Siehe xray in der AWS CLI Befehlsreferenz
- SDKs — Die entsprechende SDK-Dokumentation finden Sie auf der AWS Dokumentationsseite

> **Note**
>
> Wenn Sie einer X-Ray-Ressource keine Tags hinzufügen oder ändern können oder wenn Sie keine Ressource mit bestimmten Tags hinzufügen können, verfügen Sie möglicherweise nicht über die erforderlichen Berechtigungen, um diesen Vorgang auszuführen. Um Zugriff zu beantragen, wenden Sie sich an einen AWS Benutzer in Ihrem Unternehmen, der über Administratorrechte in X-Ray verfügt.

Themen

Tag-Einschränkungen

Für Tags gelten die folgenden Einschränkungen.

- Maximale Anzahl von Tags pro Ressource: 50
- Maximale Schlüssellänge – 128 Unicode-Zeichen.
- Maximale Wertlänge – 256 Unicode-Zeichen.
- Gültige Werte für Schlüssel- und Wert sind a-z, A-Z, 0-9, Leerzeichen und die folgenden Zeichen: _ . : / = + - und @.
- Bei Tag-Schlüsseln und -Werten muss die Groß- und Kleinschreibung beachtet werden.
- Verwenden Sie es nicht `aws:` als Präfix für Schlüssel; es ist für die AWS Verwendung reserviert.

> **Note**
>
> Sie können Systemtags nicht bearbeiten oder löschen.

Tags in der Konsole verwalten

Sie können optionale Tags hinzufügen, wenn Sie eine X-Ray-Gruppe oder eine Probenahmeregel erstellen. Tags können auch später in der Konsole geändert oder gelöscht werden.

Die folgenden Verfahren erklären, wie Sie Tags für Ihre Gruppen und Sampling-Regeln in der X-Ray-Konsole hinzufügen, bearbeiten und löschen.

Themen

Fügen Sie Tags zu einer neuen Gruppe (Konsole) hinzu

Wenn Sie eine neue X-Ray-Gruppe erstellen, können Sie optionale Tags auf der Seite Gruppe erstellen hinzufügen.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Erweitern Sie im Navigationsbereich die Option Konfiguration und wählen Sie Gruppen aus.
3. Wählen Sie Create group (Gruppe erstellen) aus.
4. Geben Sie auf der Seite Gruppe erstellen einen Namen und einen Filterausdruck für die Gruppe an. Weitere Informationen zu diesen Eigenschaften finden Sie unter Gruppen konfigurieren.
5. Geben Sie im Feld Tags einen Tag-Schlüssel und optional einen Tag-Wert ein. Sie können beispielsweise den Tag-Schlüssel und den Tag-Wert von **`Stage`** eingeben**`Production`**, um anzugeben, dass diese Gruppe für Produktionszwecke bestimmt ist. Wenn Sie ein Tag hinzufügen, wird eine neue Zeile angezeigt, in der Sie bei Bedarf ein weiteres Tag hinzufügen können. Tag-EinschränkungenIn diesem Thema finden Sie Einschränkungen für Tags.
6. Wenn Sie mit dem Hinzufügen von Tags fertig sind, wählen Sie Gruppe erstellen.

Fügen Sie Stichwörter zu einer neuen Stichprobenregel hinzu (Konsole)

Wenn Sie eine neue Röntgenprobenregel erstellen, können Sie auf der Seite Probenahmeregel erstellen Tags hinzufügen.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Erweitern Sie im Navigationsbereich die Option Konfiguration und wählen Sie Sampling aus.
3. Wählen Sie Stichprobenregel erstellen aus.
4. Geben Sie auf der Seite Stichprobenregel erstellen einen Namen, eine Priorität, Grenzwerte, Übereinstimmungskriterien und passende Attribute an. Weitere Informationen zu diesen Eigenschaften finden Sie unter Konfigurieren von -Samplingregeln.
5. Geben Sie im Feld Tags einen Tag-Schlüssel und optional einen Tag-Wert ein. Sie können beispielsweise den Tag-Schlüssel und den Tag-Wert von eingeben**`Stage`**, um anzugeben**`Production`**, dass diese Stichprobenregel für Produktionszwecke bestimmt ist. Wenn Sie ein Tag hinzufügen, wird eine neue Zeile angezeigt, in der Sie bei Bedarf ein weiteres Tag hinzufügen können. Tag-EinschränkungenIn diesem Thema finden Sie Einschränkungen für Tags.
6. Wenn Sie mit dem Hinzufügen von Tags fertig sind, wählen Sie Stichprobenregel erstellen aus.

Bearbeiten oder löschen Sie Tags für eine Gruppe (Konsole)

Sie können Tags in einer X-Ray-Gruppe auf der Seite Gruppe bearbeiten ändern oder löschen.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Erweitern Sie im Navigationsbereich die Option Konfiguration und wählen Sie Gruppen aus.
3. Wählen Sie in der Tabelle Gruppen den Namen einer Gruppe aus.
4. Bearbeiten Sie auf der Seite Gruppe bearbeiten unter Tags die Tag-Schlüssel und -Werte. Sie können keine doppelten Tag-Schlüssel haben. Tag-Werte sind optional; Sie können Werte löschen, falls gewünscht. Weitere Informationen zu anderen Eigenschaften auf der Seite Gruppe bearbeiten finden Sie unterGruppen konfigurieren. Tag-EinschränkungenIn diesem Thema finden Sie Einschränkungen für Tags.
5. Um ein Tag zu löschen, wählen Sie X rechts neben dem Tag aus.

6. Wenn Sie mit dem Bearbeiten oder Löschen von Tags fertig sind, wählen Sie Gruppe aktualisieren.

Bearbeiten oder löschen Sie Tags für eine Stichprobenregel (Konsole)

Sie können Tags in einer Röntgen-Sampling-Regel auf der Seite Sampling-Regel bearbeiten ändern oder löschen.

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die X-Ray-Konsole zu https://console.aws.amazon.com/xray/Hause.
2. Erweitern Sie im Navigationsbereich die Option Konfiguration und wählen Sie Sampling aus.
3. Wählen Sie in der Tabelle mit den Stichprobenregeln den Namen einer Stichprobenregel aus.
4. Bearbeiten Sie unter Tags die Tag-Schlüssel und -Werte. Sie können keine doppelten Tag-Schlüssel haben. Tag-Werte sind optional; Sie können Werte löschen, falls gewünscht. Weitere Informationen zu anderen Eigenschaften auf der Seite Stichprobenregel bearbeiten finden Sie unterKonfigurieren von -Samplingregeln. Tag-EinschränkungenIn diesem Thema finden Sie Einschränkungen für Tags.
5. Um ein Tag zu löschen, wählen Sie X rechts neben dem Tag aus.
6. Wenn Sie mit dem Bearbeiten oder Löschen von Tags fertig sind, wählen Sie Stichprobenregel aktualisieren.

Verwaltung von Stichwörtern in AWS CLI

Sie können Tags hinzufügen, wenn Sie eine X-Ray-Gruppe oder eine Probenahmeregel erstellen. Sie können die auch verwenden AWS CLI , um Tags zu erstellen und zu verwalten. Um Tags für eine bestehende Gruppe oder Stichprobenregel zu aktualisieren, verwenden Sie die AWS X-Ray Konsole oder das TagResourceoder UntagResource APIs.

Themen

- Hinzufügen von Tags zu einer neuen X-Ray-Gruppe oder Sampling-Regel (CLI)
- Hinzufügen von Tags zu einer vorhandenen Ressource (CLI)
- Tags auf einer Ressource auflisten (CLI)
- Tags auf einer Ressource löschen (CLI)

Hinzufügen von Tags zu einer neuen X-Ray-Gruppe oder Sampling-Regel (CLI)

Verwenden Sie einen der folgenden Befehle, um beim Erstellen einer neuen X-Ray-Gruppe oder Sampling-Regel optionale Tags hinzuzufügen.

- Um einer neuen Gruppe Tags hinzuzufügen, führen Sie den folgenden Befehl aus und *group_name* ersetzen Sie ihn durch den Namen Ihrer Gruppe, *mydomain.com* durch den Endpunkt Ihres Dienstes, *key_name* durch einen Tag-Schlüssel und optional *value* durch einen Tag-Wert. Weitere Informationen zum Erstellen einer Gruppe finden Sie unterGruppen.

  ```
  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"}]
  ```

  Im Folgenden wird ein Beispiel gezeigt.

  ```
  aws xray create-group \
     --group-name "AdminGroup" \
     --filter-expression "service(\"mydomain.com\") {fault OR error}" \
     --tags [{"Key": "Stage","Value": "Prod"},{"Key": "Department","Value": "QA"}]
  ```

- Um einer neuen Stichprobenregel Tags hinzuzufügen, führen Sie den folgenden Befehl aus und *key_name* ersetzen Sie ihn durch einen Tag-Schlüssel und optional *value* durch einen Tag-Wert. Dieser Befehl gibt die Werte im `--sampling-rule` Parameter als JSON-Datei an. Weitere Informationen zum Erstellen einer Stichprobenregel finden Sie unterSamplingregeln.

  ```
  aws xray create-sampling-rule \
     --cli-input-json file://file_name.json
  ```

  Im Folgenden finden Sie den Inhalt der JSON-Dateifile_name.json, die durch den `--cli-input-json` Parameter angegeben wird.

  ```
  {
      "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"
           }
          ]
}
```

Nachfolgend finden Sie einen Beispielbefehl.

```
aws xray create-sampling-rule \
   --cli-input-json file://9000-base-scorekeep.json
```

Im Folgenden finden Sie den Inhalt der durch den `--cli-input-json` Parameter angegebenen `9000-base-scorekeep.json` Beispieldatei.

```
{
    "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"
           }
          ]
}
```

Hinzufügen von Tags zu einer vorhandenen Ressource (CLI)

Sie können den `tag-resource` Befehl ausführen, um einer vorhandenen X-Ray-Gruppe oder einer Stichprobenregel Tags hinzuzufügen. Diese Methode ist möglicherweise einfacher als das Hinzufügen von Tags durch Ausführen von `update-group` oder`update-sampling-rule`.

Um einer Gruppe oder einer Stichprobenregel Tags hinzuzufügen, führen Sie den folgenden Befehl aus. Ersetzen Sie dabei den ARN durch den ARN der Ressource und geben Sie die Schlüssel und optionalen Werte der Tags an, die Sie hinzufügen möchten.

```
aws xray tag-resource \
   --resource-arn "ARN" \
   --tag-keys [{"Key":"key_name","Value":"value"}, {"Key":"key_name","Value":"value"}]
```

Im Folgenden wird ein Beispiel gezeigt.

```
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"}]
```

Tags auf einer Ressource auflisten (CLI)

Sie können den `list-tags-for-resource` Befehl ausführen, um die Tags einer X-Ray-Gruppe oder einer Stichprobenregel aufzulisten.

Um die Tags aufzulisten, die einer Gruppe oder einer Stichprobenregel zugeordnet sind, führen Sie den folgenden Befehl aus und ersetzen Sie den ARN durch den ARN der Ressource.

```
aws xray list-tags-for-resource \
   --resource-arn "ARN"
```

Im Folgenden wird ein Beispiel gezeigt.

```
aws xray list-tags-for-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup"
```

Tags auf einer Ressource löschen (CLI)

Sie können den `untag-resource` Befehl ausführen, um Tags aus einer X-Ray-Gruppe oder einer Stichprobenregel zu entfernen.

Um Tags aus einer Gruppe oder einer Stichprobenregel zu entfernen, führen Sie den folgenden Befehl aus. Ersetzen Sie dabei den ARN durch den ARN der Ressource und geben Sie die Schlüssel der Tags an, die Sie entfernen möchten.

Mit dem `untag-resource` Befehl können Sie nur ganze Tags entfernen. Um Tag-Werte zu entfernen, verwenden Sie die X-Ray-Konsole oder löschen Sie Tags und fügen Sie neue Tags mit denselben Schlüsseln, aber unterschiedlichen oder leeren Werten hinzu.

```
aws xray untag-resource \
   --resource-arn "ARN" \
   --tag-keys ["key_name","key_name"]
```

Im Folgenden wird ein Beispiel gezeigt.

```
aws xray untag-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/group_name" \
   --tag-keys ["Stage","Department"]
```

Steuern Sie den Zugriff auf X-Ray-Ressourcen anhand von Tags

Sie können Tags an X-Ray-Gruppen oder Sampling-Regeln anhängen oder Tags in einer Anfrage an X-Ray übergeben. Um den Zugriff auf der Grundlage von Tags zu steuern, geben

Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `xray:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden. Weitere Informationen zu diesen Bedingungsschlüsseln finden Sie unter Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags.

Ein Beispiel für eine identitätsbasierte Richtlinie zur Einschränkung des Zugriffs auf eine Ressource auf der Grundlage der Markierungen dieser Ressource finden Sie unter Verwaltung des Zugriffs auf X-Ray-Gruppen und Stichprobenregeln auf der Grundlage von Tags.

Problembehebung AWS X-Ray

In diesem Thema werden häufig auftretende Fehler und Probleme aufgeführt, die bei der Verwendung der X-Ray-API, der X-Ray-Konsole oder auftreten können SDKs. Wenn Sie auf ein Problem stoßen, das hier nicht aufgeführt ist, können Sie die Schaltfläche Feedback auf dieser Seite verwenden, um es zu melden.

Sections

Seiten mit Röntgen-Trace-Map und Trace-Details

Die folgenden Abschnitte können Ihnen helfen, wenn Sie Probleme mit der X-Ray-Trace-Map und der Trace-Detailseite haben:

Ich sehe nicht alle meine CloudWatch Logs

Wie Logs so konfiguriert werden, dass sie auf den Seiten mit der X-Ray-Trace-Map und den Trace-Details angezeigt werden, hängt vom jeweiligen Dienst ab.

- API-Gateway-Protokolle werden angezeigt, wenn die Protokollierung in API Gateway aktiviert ist.

Nicht alle Service Map-Knoten unterstützen die Anzeige der zugehörigen Protokolle. Logs für die folgenden Knotentypen anzeigen:

- Lambda-Kontext
- Lambda-Funktion
- API-Gateway-Phase
- Amazon-ECS-Cluster
- Amazon ECS-Instanz
- Amazon-ECS-Service

- Amazon-ECS-Aufgaben
- Amazon-EKS-Cluster
- Amazon EKS-Namespace
- Amazon EKS-Knoten
- Amazon EKS-Pod
- Amazon EKS-Dienst

Ich sehe nicht alle meine Alarme auf der X-Ray-Trace-Map

Die X-Ray-Trace-Map zeigt nur das Warnsymbol für einen Knoten an, wenn sich Alarme, die diesem Knoten zugeordnet sind, im ALARM-Status befinden.

Die Trace-Map ordnet Alarme mithilfe der folgenden Logik Knoten zu:

- Wenn der Knoten einen AWS Dienst darstellt, sind alle Alarme, deren Namespace diesem Dienst zugeordnet ist, diesem Knoten zugeordnet. Beispielsweise `AWS::Kinesis` ist ein Knoten des Typs mit allen Alarmen verknüpft, die auf Metriken im CloudWatch `AWS/Kinesis` Namespace basieren.
- Wenn der Knoten eine AWS Ressource darstellt, sind die Alarme für diese spezifische Ressource verknüpft. Beispielsweise ist ein Knoten vom Typ `AWS::DynamoDB::Table` mit dem Namen „MyTable“ mit allen Alarmen verknüpft, die auf einer Metrik mit dem Namespace basieren `AWS/DynamoDB` und deren `TableName` Dimension auf `MyTable` gesetzt ist.
- Wenn der Knoten vom unbekannten Typ ist, der durch einen gestrichelten Rahmen um den Namen identifiziert wird, sind diesem Knoten keine Alarme zugeordnet.

Ich sehe einige AWS Ressourcen nicht auf der Trace-Map

Nicht jede AWS Ressource wird durch einen dedizierten Knoten repräsentiert. Einige AWS Dienste werden durch einen einzigen Knoten für alle Anfragen an den Dienst repräsentiert. Die folgenden Ressourcentypen werden mit einem Knoten pro Ressource angezeigt:

- `AWS::DynamoDB::Table`
- `AWS::Lambda::Function`

  Lambda-Funktionen werden durch zwei Knoten dargestellt — einen für den Lambda-Container und einen für die Funktion. Dies hilft, Kaltstartprobleme mit Lambda-Funktionen zu identifizieren.

Lambda-Container-Knoten werden auf dieselbe Weise mit Alarmen und Dashboards verknüpft wie Lambda-Funktionsknoten.

- `AWS::ApiGateway::Stage`
- `AWS::SQS::Queue`
- `AWS::SNS::Topic`

Die Trace-Map enthält zu viele Knoten

Verwenden Sie X-Ray-Gruppen, um Ihre Übersicht in mehrere Übersichten aufzuteilen. Weitere Informationen finden Sie unter Verwenden von Filterausdrücken mit Gruppen.

X-Ray SDK for Java

Fehler: Ausnahme im Thread „Thread-1" com.amazonaws.xray.exceptions.SegmentNotFoundException: Das Untersegment mit dem Namen 'AmazonSNS' konnte nicht gestartet werden: Das Segment konnte nicht gefunden werden.

Dieser Fehler weist darauf hin, dass das X-Ray-SDK versucht hat, einen ausgehenden Anruf aufzuzeichnen AWS, aber kein offenes Segment finden konnte. Dies kann in folgenden Situationen auftreten:

- Ein Servlet-Filter ist nicht konfiguriert — Das X-Ray SDK erstellt Segmente für eingehende Anfragen mit einem Filter namens`AWSXRayServletFilter`. Konfigurieren Sie einen Servlet-Filter, um eingehende Anfragen zu instrumentieren.
- Sie verwenden instrumentierte Clients außerhalb des Servlet-Codes — Wenn Sie einen instrumentierten Client verwenden, um Aufrufe in Startcode oder anderem Code zu tätigen, der nicht als Antwort auf eine eingehende Anfrage ausgeführt wird, müssen Sie ein Segment manuell erstellen. Beispiele finden Sie unter Instrumentieren von Startup-Code.
- Sie verwenden instrumentierte Clients in Worker-Threads. Wenn Sie einen neuen Thread erstellen, verliert der X-Ray-Recorder seinen Verweis auf das offene Segment. Sie können die Methoden `getTraceEntity` und `setTraceEntity` verwenden, um eine Referenz zum aktuellen Segment oder Untersegment zu erhalten (`Entity`) und sie innerhalb des Threads wieder an den Recorder zu übergeben. Ein Beispiel finden Sie unter Verwenden instrumentierter Clients in Auftragnehmer-Threads.

X-Ray-SDK für Node.js

Problem: CLS funktioniert nicht mit Sequelize

Übergeben Sie das X-Ray-SDK für den Namespace Node.js mit der Methode an Sequelize. `cls`

```
var AWSXRay = require('aws-xray-sdk');
const Sequelize = require('sequelize');
Sequelize.cls = AWSXRay.getNamespace();
const sequelize = new Sequelize(...);
```

Problem: CLS funktioniert nicht mit Bluebird

Verwenden Sie `cls-bluebird`, damit Bluebird mit CLS funktioniert.

```
var AWSXRay = require('aws-xray-sdk');
var Promise = require('bluebird');
var clsBluebird = require('cls-bluebird');
clsBluebird(AWSXRay.getNamespace());
```

Der X-Ray-Daemon

Problem: Der Daemon verwendet die falschen Anmeldeinformationen

Der Daemon verwendet das AWS SDK, um Anmeldeinformationen zu laden. Wenn Sie Anmeldeinformationen mit mehreren Methoden bereitstellen, wird die Methode mit der höchsten Priorität verwendet. Weitere Informationen finden Sie unter Ausführen des Daemons.

Dokumentenverlauf für AWS X-Ray

In der folgenden Tabelle werden die wichtigen Änderungen an der Dokumentation für AWS X-Ray beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Letzte Aktualisierung der Dokumentation: 07. März 2024

Änderung	Beschreibung	Datum
Aktualisierter Zeitplan für die Unterstützung von AWS X-Ray SDKs und Daemon	Am 25. Februar 2026 wechselt das AWS X-Ray SDKs/Daemon in den Wartungsmodus, in dem die Versionen von X-Ray SDK und Daemon auf Sicherheitsprobleme beschränkt AWS werden. Weitere Informationen zur Support-Zeitleiste finden Sie unter X-Ray SDK und Daemon Support Timeline. Wir empfehlen die Migration zu OpenTelemetry. Weitere Informationen zur Migration zu OpenTelemetry finden Sie unter Migration von X-Ray-Instrumentierung zu OpenTelemetry Instrumentierung.	26. November 2025
end-of-supportHinweis für AWS X-Ray SDKs und Daemon hinzugefügt	Am 25. Februar 2027 wird AWS X-Ray die Unterstützung für AWS X-Ray SDKs und Daemon einstellen. Wir empfehlen die Migration zu. OpenTelemetry Weitere Informationen finden Sie unter Migration von Röntgenin	22. August 2025

	strumenten zur OpenTelemetry Instrumentierung.	
Hinzugefügte Funktionalität	Migration von X-Ray zu OpenTelemetry. Weitere Informationen finden Sie unter Migration von Röntgeninstrumenten zur OpenTelemetry Instrumentierung.	13. Juni 2025
Hinzugefügte Funktionalität	AWS X-Ray unterstützt jetzt die Transaktionssuche. Weitere Informationen finden Sie unter Transaktionssuche.	21. November 2024
Hinzugefügte Funktionalität	AWS X-Ray unterstützt jetzt OpenTelemetry Protocol (OTLP) Endpoint. Weitere Informationen finden Sie unter OpenTelemetry.	21. November 2024
Hinzugefügte Funktionalität	X-Ray protokolliert jetzt Datenereignisse `PutTraceSegments` , einschließlich `GetTraceSummaries` , und `BatchGetTraces` zu AWS CloudTrail. X-Ray protokolliert jetzt auch das `GetSamplingStatisticSummaries` Verwaltungsereignis unter CloudTrail. Weitere Informationen finden Sie unter Protokollieren von X-Ray-API-Aufrufen mit AWS CloudTrail.	7. März 2024

Hinzugefügte Funktionalität	X-Ray unterstützt jetzt Trace, die mit OpenTelemetry oder einem anderen Framework IDs erstellt wurden, das der W3C Trace Context-Spezifikation entspricht. Weitere Informati onen finden Sie unter Trace-Daten an X-Ray senden.	25. Oktober 2023
Hinzugefügte Funktionalität	Sie können jetzt das aktive Tracing von Amazon SNS konfigurieren, sodass Sie Anfragen verfolgen und analysieren können, während sie Ihre Amazon SNS SNS-Themen durchlaufen. Weitere Informationen finden Sie unter Amazon SNS und AWS X-Ray.	8. Februar 2023
Thema „X-Ray SDK für Node.js“ aktualisiert	Es wurden Details zur Instrumentierung von Clients hinzugefügt, die AWS SDK für JavaScript V3 verwenden . Einzelheiten finden Sie unter Nachverfolgen von AWS SDK-Aufrufen mit dem X-Ray-SDK für Node.js.	07. Februar 2023

Die Einzelheiten der von IAM verwalteten Richtlinie wurden aktualisiert	Den Richtlinien und verwaltet en Richtlinien wurde eine IAM-Berechtigung für kontoübergreifende Beobachtbarkeit hinzugefü gt. `AWSXRayReadOnlyAcc ess` `AWSXRayFullAccess` `AWSXrayCrossAccoun tSharingConfigurat ion` Einzelheiten finden Sie unter IAM-verwaltete Richtlini en für X-Ray.	07. Februar 2023
Hinzugefügte Funktionalität	AWS X-Ray unterstützt jetzt kontenübergreifende Observability, sodass Sie Anwendungen überwachen und Fehler beheben können, die sich über mehrere Konten innerhalb eines AWS-Region Einzelheiten finden Sie unter Kontoübergreifende Ablaufver folgung.	27. November 2022

Hinzugefügte Funktionalität	Sie können jetzt verknüpfte Traces zwischen Nachrichtenproduzenten, einer Amazon SQS SQS-Warteschlange und Verbrauchern anzeigen und so eine verbundene Ansicht der Traces bieten, die von ereignisgesteuerten Anwendungen gesendet wurden. Weitere Informationen finden Sie unter Ablaufverfolgung ereignisgesteuerter Anwendungen.	20. November 2022
Die Informationen zu den von IAM verwalteten Richtlinien wurden aktualisiert	Die IAM-Berechtigung zum Auflisten von Ressourcenrichtlinien wurde zur `AWSXRayReadOnlyAccess` verwalteten Richtlinie hinzugefügt. Einzelheiten finden Sie unter IAM-verwaltete Richtlinien für X-Ray.	15. November 2022
Die Berechtigungen der IAM-Konsole und die Details der verwalteten Richtlinien wurden aktualisiert	Der Satz der IAM-Berechtigungen, die die X-Ray-Konsole verwendet, wurde zusammen mit der Beschreibung der `AWSXRayReadOnlyAccess` verwalteten Richtlinie aktualisiert. Einzelheiten finden Sie unter Verwenden der X-Ray-Konsole.	11. November 2022

AWS Distribution für OpenTelemetry Ruby hinzugefügt	AWS Distro for OpenTelemetry (ADOT) bietet einen einzigen Satz von Open Source APIs, Bibliotheken und Agenten zum Sammeln verteilter Traces und Metriken. ADOT Ruby ermöglicht es Ihnen, Ihre Ruby-Anwendung für X-Ray und andere Tracing-Backends zu instrumentieren. Weitere Informationen finden Sie unter AWS Distro for Ruby. OpenTelemetry	7. Februar 2022
Hinzugefügte Funktionalität	Sie können jetzt Traces anzeigen und X-Ray von der CloudWatch Konsole aus konfigurieren. Weitere Informationen finden Sie unter X-Ray-Konsole.	24. Januar 2022
Integriertes CloudWatch RUM	Mit AWS X-Ray und CloudWatch RUM können Sie den Anforderungspfad analysieren und debuggen, angefangen bei den Endbenutzern Ihrer Anwendung bis hin zu nachgelagerten AWS Managed Services. Weitere Informationen finden Sie unter CloudWatch RUM und AWS X-Ray.	3. Dezember 2021

Integrierte AWS Distribution für OpenTelemetry	The AWS Distro for OpenTelemetry (ADOT) bietet einen einzigen Satz von Open Source APIs, Bibliotheken und Agenten zur Erfassung verteilter Traces und Metriken. ADOT ermöglicht es Ihnen, Ihre Anwendung für X-Ray und andere Tracing-Backends zu instrumentieren. Weitere Informationen finden Sie unter Instrumentierung Ihrer App.	23. September 2021
Hinzugefügte Funktionalität	AWS X-Ray lässt sich jetzt in Amazon Virtual Private Cloud integrieren, sodass Ressourcen in Ihrer Amazon VPC mit dem X-Ray-Service kommunizieren können, ohne das öffentliche Internet nutzen zu müssen. Weitere Informationen finden Sie unter Verwenden AWS X-Ray mit VPC-Endpunkten.	20. Mai 2021
Hinzugefügte Funktionalität	AWS X-Ray lässt sich jetzt in integrieren CloudFormation und ermöglicht Ihnen die Bereitstellung und Konfiguration von X-Ray-Ressourcen. Weitere Informationen finden Sie unter X-Ray-Ressourcen erstellen mit CloudFormation.	6. Mai 2021

Hinzugefügte Funktionalität	AWS X-Ray lässt sich jetzt in Amazon integrieren EventBrid ge , um Ereignisse nachzuver folgen, die durchlaufen wurden EventBridge. Dies bietet Benutzern einen vollständ igeren Überblick über ihr System. Weitere Informati onen finden Sie unter Amazon EventBridge und AWS X-Ray.	2. März 2021
Daemon zu ECR hinzugefügt	Der Daemon kann jetzt von Amazon ECR heruntergeladen werden. Weitere Informationen finden Sie unter Den Daemon herunterladen.	1. März 2021
Hinzugefügte Funktionalität	AWS X-Ray unterstützt jetzt Insights-bezogene Benachrichtigungen an Amazon EventBridge. Auf diese Weise können Sie mithilfe von Insights automatis che Maßnahmen ergreifen EventBridge. Weitere Informati onen finden Sie unter Insights-Benachrichtigungen.	15. Oktober 2020
Herunterladbare Daemons hinzugefügt	AWS X-Ray führt einen Support-Daemon für Linux ein. ARM64 Weitere Informationen finden Sie unter AWS X-Ray daemonbrazil ws	1. Oktober 2020

Hinzugefügte Funktionalität	AWS X-Ray unterstützt jetzt die aktive Integration mit Amazon CloudWatch Synthetics. Auf diese Weise können Sie Details zu einem Synthetics Canary-Client-Knoten wie Antwortzeit und Status anzeigen. Sie können in der Analytics-Konsole auch Analysen auf der Grundlage von Informationen aus einem Canary-Clientknote n von Synthetics durchführ en. Weitere Informationen finden Sie unter Debuggen CloudWatch synthetischer Kanarien mit X-Ray.	24. September 2020
Hinzugefügte Funktionalität	AWS X-Ray unterstützt jetzt end-to-end Ablaufver folgungs-Workflows für. AWS Step Functions Sie können die Komponenten Ihrer Zustandsmaschine visualisi eren, Leistungsengpässe identifizieren und Anfragen, die zu einem Fehler geführt haben, beheben. Weitere Informationen finden Sie unter AWS Step Functions und. AWS X-Ray	14. September 2020

Hinzugefügte Funktionalität	AWS X-Ray bietet Einblicke zur kontinuierlichen Analyse von Trace-Daten in Ihrem Konto, um aufkommende Probleme in Ihren Anwendungen zu identifizieren. Insights zeichnet Vorfälle auf und verfolgt die Auswirkungen von Vorfällen bis zur Lösung. Weitere Informationen finden Sie unter Insights in der AWS X-Ray Konsole verwenden	3. September 2020
Hinzugefügte Funktionalität	AWS X-Ray führt den Java-Agenten für automatische Instrumentierung ein, der es Kunden ermöglicht, Trace-Daten zu sammeln, ohne die bestehende Java-basierte Anwendung ändern zu müssen. Sie können jetzt Web- und Servlet-basierte Java-Anwendungen mit minimalen Konfigurationsänderungen und ohne Codeänderungen verfolgen. Weitere Informationen finden Sie unter AWS X-Ray Auto-Instrumentierungs-Agent für Java.	3. September 2020

Hinzugefügte Funktionalität	AWS X-Ray hat der X-Ray-Konsole eine neue Gruppenseite hinzugefügt, um die Erstellung und Verwaltung von Trace-Gruppen zu vereinfachen. Weitere Informationen finden Sie unter Gruppen in der X-Ray-Konsole konfigurieren.	24. August 2020
Hinzugefügte Funktionalität	AWS X-Ray ermöglicht jetzt das Hinzufügen von Tags zu Gruppen und Sampling-Regeln. Sie können auch den Zugriff auf Gruppen und Stichprobenregeln anhand von Stichwörtern steuern. Weitere Informationen finden Sie unter Kennzeichnen von Regeln und Gruppen für X-Ray und Verwaltung des Zugriffs auf Röntgengruppen und Stichprobenregeln auf der Grundlage von Tags.	24. August 2020

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.