



Guia do Desenvolvedor

AWS X-Ray



AWS X-Ray: Guia do Desenvolvedor

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não são propriedade da Amazon pertencem aos respectivos proprietários, os quais podem ou não ser afiliados, estar conectados ou ser patrocinados pela Amazon.

Table of Contents

O que é o AWS X-Ray?	1
Introdução	4
Escolher uma interface	7
Usar um SDK	9
Usar o SDK do ADOT	10
Usar o SDK do X-Ray	11
Usar um console	13
Usar o console do Amazon CloudWatch	13
Usar o console do X-Ray	14
Explorar o console do X-Ray	15
Mapa de rastreamento	16
Rastreamentos	23
Expressões de filtro	32
Rastreamento entre contas	45
Rastrear aplicações orientadas a eventos	49
Histogramas	53
Informações	56
Analytics	64
Grupos	71
Amostragem	81
Amostragem adaptável	88
Vinculação direta do console	97
Usar a API do X-Ray	100
Tutorial	102
Envio de dados	107
Obter dados	112
Configuração	127
Amostragem	134
Documentos de segmento	138
Conceitos	159
Segmentos	159
Subsegmentos	160
Gráfico de serviço	164
Rastreamentos	165

Amostragem	166
Cabeçalho de rastreamento	167
Expressões de filtro	168
Grupos	169
Anotações e metadados	170
Erros, falhas e exceções	170
Segurança	172
Proteção de dados	172
Gerenciamento de identidade e acesso	175
Público	175
Autenticação com identidades	176
Gerenciar o acesso usando políticas	177
Como o AWS X-Ray funciona com o IAM	179
Exemplos de políticas baseadas em identidade	186
Solução de problemas	198
Registrar em log e monitoramento	200
Validação de conformidade	201
Resiliência	201
Segurança da infraestrutura	202
Endpoints da VPC	202
Criar um endpoint da VPC para o X-Ray	203
Controlar o acesso ao endpoint da VPC do X-Ray	204
Regiões com suporte	205
Prevenção contra o ataque do “substituto confuso” em todos os serviços	206
Aplicação de exemplo	209
Tutorial do Scorekeep	212
Pré-requisitos	213
Instale o aplicativo Scorekeep usando CloudFormation	214
Gerar dados de rastreamento	215
Veja o mapa de rastreamento no Console de gerenciamento da AWS	216
Configuração de notificações do Amazon SNS	224
Explorar o aplicativo de amostra	226
Opcional: política de menor privilégio	231
Limpeza	233
Próximas etapas	234
AWS Clientes SDK	235

Subsegmentos personalizados	236
Anotações e metadados	237
Clientes HTTP	238
Clientes SQL	239
AWS Lambda funções	242
Nome aleatório	243
Operador	245
Instrumentar código de inicialização	248
Scripts de instrumentação	250
Instrumentar clientes web	252
Threads de operador	256
Daemon do X-Ray	259
Download do daemon	260
Verificar a assinatura de arquivamento do daemon	261
Execução do daemon	262
Conceder permissão ao daemon para enviar dados ao X-Ray	263
Logs do daemon do X-Ray	264
Configuração	264
Variáveis de ambiente compatíveis	265
Usar as opções de linha de comando	265
Usar um arquivo de configuração	267
Executar o daemon localmente	268
Executar o daemon do X-Ray no Linux	269
Executar o daemon do X-Ray em um contêiner do Docker	269
Executar o daemon do X-Ray-Ray no Windows	271
Executar o daemon do X-Ray no OS X	272
No Elastic Beanstalk	272
Usar a integração entre do X-Ray com o Elastic Beanstalk para executar o daemon do X-Ray	273
Baixar e executar o daemon do X-Ray manualmente (avançado)	275
Na Amazon EC2	277
No Amazon ECS	278
Usar a imagem oficial do Docker da	279
Criar e compilar uma imagem do Docker	279
Configurar opções da linha de comandos no console do Amazon ECS	282
Integrando com Serviços da AWS	284

Amazon Bedrock AgentCore	286
Amazon S3	287
Amazon S3	287
Amazon EC2	287
Amazon SNS	288
Configurar o rastreamento ativo do Amazon SNS	288
Visualize rastreamentos de publicadores e assinantes do Amazon SNS no console do X-Ray	290
Amazon SQS	292
Enviar o cabeçalho de rastreamento HTTP	293
Recuperar o cabeçalho de rastreamento e recuperar o contexto de rastreamento	294
Amazon S3	295
Configurar notificações de eventos do Amazon S3	296
AWS Distro para OpenTelemetry	297
AWS Distro para OpenTelemetry	297
AWS Config	298
Criar um acionador de função do Lambda	298
Criar uma regra do AWS Config personalizada para o X-Ray	300
Resultados de exemplo	301
Notificações do Amazon SNS	301
AWS AppSync	301
API Gateway	302
App Mesh	304
App Runner	307
CloudTrail	307
Eventos de gerenciamento no CloudTrail	309
Eventos de dados no CloudTrail	310
Exemplos de eventos X-Ray	311
CloudWatch	314
CloudWatch RUM	314
CloudWatch Synthetics	316
Elastic Beanstalk	325
ELB	326
EventBridge	327
Visualizar a origem e os destinos no mapa de serviço do X-Ray	327
Propagar o contexto de rastreamento para destinos de eventos	328

Lambda	334
Step Functions	336
Instrumentar seu aplicativo	338
Instrumentando seu aplicativo com a AWS Distro para OpenTelemetry	339
Instrumentando seu aplicativo com AWS X-Ray SDKs	340
Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs	341
Transaction Search	343
OpenTelemetry Ponto final do protocolo (OTLP)	344
Trabalho com Go	345
AWS Distro para Go OpenTelemetry	345
X-Ray SDK para Go	346
Requisitos	348
Documentação de referência	348
Configuração	348
Solicitações de entrada	355
AWS Clientes SDK	359
Chamadas HTTP de saída	361
Consultas SQL	362
Subsegmentos personalizados	362
Anotações e metadados	363
Como trabalhar com Java	367
AWS Distro para OpenTelemetry Java	367
X-Ray SDK para Java	368
Submódulos	370
Requisitos	371
Gerenciamento de dependências	371
Agente de instrumentação automática	373
Configuração	387
Solicitações de entrada	399
AWS Clientes SDK	404
Chamadas HTTP de saída	407
Consultas SQL	409
Subsegmentos personalizados	412
Anotações e metadados	415
Monitoramento	420
Multithreading	424

AOP com Spring	426
Trabalho com Node.js	431
AWS Distro para OpenTelemetry JavaScript	431
X-Ray SDK para Node.js	432
Requisitos	434
Gerenciamento de dependências	435
Amostras de Node.js	436
Configuração	436
Solicitações de entrada	442
AWS Clientes SDK	446
Chamadas HTTP de saída	450
Consultas SQL	452
Subsegmentos personalizados	453
Anotações e metadados	456
Como trabalhar com Python	461
AWS Distro para OpenTelemetry Python	461
X-Ray SDK para Python	462
Requisitos	465
Gerenciamento de dependências	465
Configuração	466
Solicitações de entrada	473
Aplicar patches a bibliotecas	479
AWS Clientes SDK	482
Chamadas HTTP de saída	483
Subsegmentos personalizados	485
Anotações e metadados	487
Instrumentar aplicativos sem servidor	491
Como trabalhar com .NET	498
AWS Distro para OpenTelemetry .NET	498
X-Ray SDK para .NET	499
Requisitos	501
Adicionar o X-Ray SDK para .NET ao aplicativo	501
Gerenciamento de dependências	501
Configuração	503
Solicitações de entrada	510
AWS Clientes SDK	515

Chamadas HTTP de saída	517
Consultas SQL	519
Subsegmentos personalizados	523
Anotações e metadados	524
Trabalhar com Ruby	528
AWS Distro para OpenTelemetry Ruby	528
X-Ray SDK para Ruby	529
Requisitos	531
Configuração	531
Solicitações de entrada	538
Aplicar patches a bibliotecas	542
AWS Clientes SDK	543
Subsegmentos personalizados	544
Anotações e metadados	545
Cronograma de suporte do X-Ray SDK e do Daemon Support	550
Migração da instrumentação X-Ray para a instrumentação OpenTelemetry	551
Compreensão OpenTelemetry	552
OpenTelemetry suporte em AWS	552
Entendendo OpenTelemetry os conceitos de migração	553
Comparando recursos	554
Configurar o rastreamento	555
Detectar recursos no seu ambiente	556
Gerenciar estratégias de amostragem	557
Gerenciar o contexto de rastreamento	558
Propagar o contexto de rastreamento	558
Usar instrumentação de biblioteca	559
Exportar rastreamentos	559
Processamento e encaminhamento de rastreamento	560
Processamento de extensão (conceito OpenTelemetry específico)	561
Bagagem (conceito específico) OpenTelemetry	561
Visão geral da migração	561
Recomendações de aplicações novas e existentes	562
Alterações na configuração de rastreamento	562
Alterações na instrumentação de biblioteca	563
Alterações na instrumentação do ambiente do Lambda	563
Criar dados de rastreamento manualmente	564

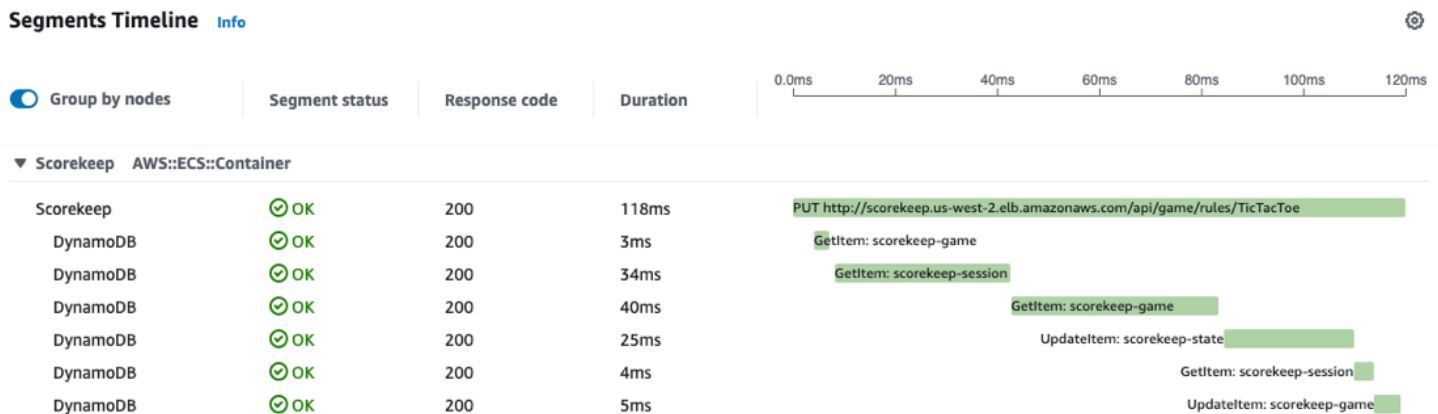
Migração do X-Ray Daemon para AWS CloudWatch agente ou coletor OpenTelemetry	564
Migração na Amazon EC2 ou em servidores locais	565
Migrar no Amazon ECS	568
Migrar no Elastic Beanstalk	573
Migrando para Java OpenTelemetry	574
Solução de instrumentação automática de código zero	574
Soluções de instrumentação manual com o SDK	575
Rastrear solicitações recebidas (instrumentação do Spring Framework)	578
AWS Instrumentação do SDK v2	579
Instrumentar chamadas HTTP de saída	580
Suporte de instrumentação para outras bibliotecas	582
Criar dados de rastreamento manualmente	582
Instrumentação do Lambda	585
Migre para Go OpenTelemetry	590
Instrumentação manual com o SDK	590
Rastrear solicitações de entrada (instrumentação do manipulador HTTP)	592
AWS SDK para instrumentação do SDK for Go v2	594
Instrumentar chamadas HTTP de saída	595
Suporte de instrumentação para outras bibliotecas	596
Criar dados de rastreamento manualmente	597
Instrumentação manual Lambda	598
Migrar para Node.js OpenTelemetry	605
Soluções de instrumentação automática de código zero	605
Soluções de instrumentação manual	606
Rastreamento de solicitações de entrada	609
AWS Instrumentação do SDK JavaScript V3	594
Instrumentar chamadas HTTP de saída	613
Suporte de instrumentação para outras bibliotecas	614
Criar dados de rastreamento manualmente	597
Instrumentação do Lambda	598
Migrar para o .NET OpenTelemetry	617
Soluções de instrumentação automática de código zero	618
Soluções de instrumentação manual com o SDK	618
Criar dados de rastreamento manualmente	622
Rastrear solicitações recebidas (instrumentação do ASP.NET e do ASP.NET Core)	624
AWS Instrumentação do SDK	625

Instrumentar chamadas HTTP de saída	626
Suporte de instrumentação para outras bibliotecas	627
Instrumentação do Lambda	598
Migrar para Python OpenTelemetry	632
Soluções de instrumentação automática de código zero	632
Instrumente suas aplicações manualmente	633
Inicialização da configuração de rastreamento	633
Rastreamento de solicitações de entrada	636
AWS Instrumentação do SDK	637
Instrumentar chamadas HTTP de saída por meio de solicitações	639
Suporte de instrumentação para outras bibliotecas	640
Criar dados de rastreamento manualmente	641
Instrumentação do Lambda	642
Migrar para Ruby OpenTelemetry	644
Instrumente manualmente suas soluções com o SDK	644
Rastrear solicitações recebidas (instrumentação Rails)	647
AWS Instrumentação do SDK	648
Instrumentar chamadas HTTP de saída	649
Suporte de instrumentação para outras bibliotecas	650
Criar dados de rastreamento manualmente	650
Instrumentação manual Lambda	652
Criar recursos do X-Ray com o CloudFormation	656
X-Ray e modelos do CloudFormation	656
Saiba mais sobre o CloudFormation	656
Tags	657
Restrições de tag	658
Gerenciar tags no console	658
Adicionar tags a um novo grupo (console)	659
Adicionar tags a uma nova regra de amostragem (console)	659
Editar ou excluir tags de um grupo (console)	660
Editar ou excluir tags de uma regra de amostragem (console)	660
Gerenciar tags na AWS CLI	661
Adicione tags a um novo grupo ou regra de amostragem do X-Ray (CLI)	661
Adicionar tags a um recurso existente (CLI)	663
Listar as tags em um recurso (CLI)	664
Excluir tags de um recurso (CLI)	664

Controlar o acesso a recursos do X-Ray com base em tags	665
Solução de problemas	666
Mapa de rastreamento do X-Ray e páginas de detalhes do rastreamento	666
Não vejo todos os meus logs do CloudWatch	666
Não vejo todos os meus alarmes no mapa de rastreamento do X-Ray	667
Não vejo alguns recursos da AWS no mapa de rastreamento	667
Há muitos nós no mapa de rastreamento	668
X-Ray SDK para Java	668
X-Ray SDK para Node.js	668
O daemon do X-Ray	669
Histórico do documento	670
.....	dclxxxi

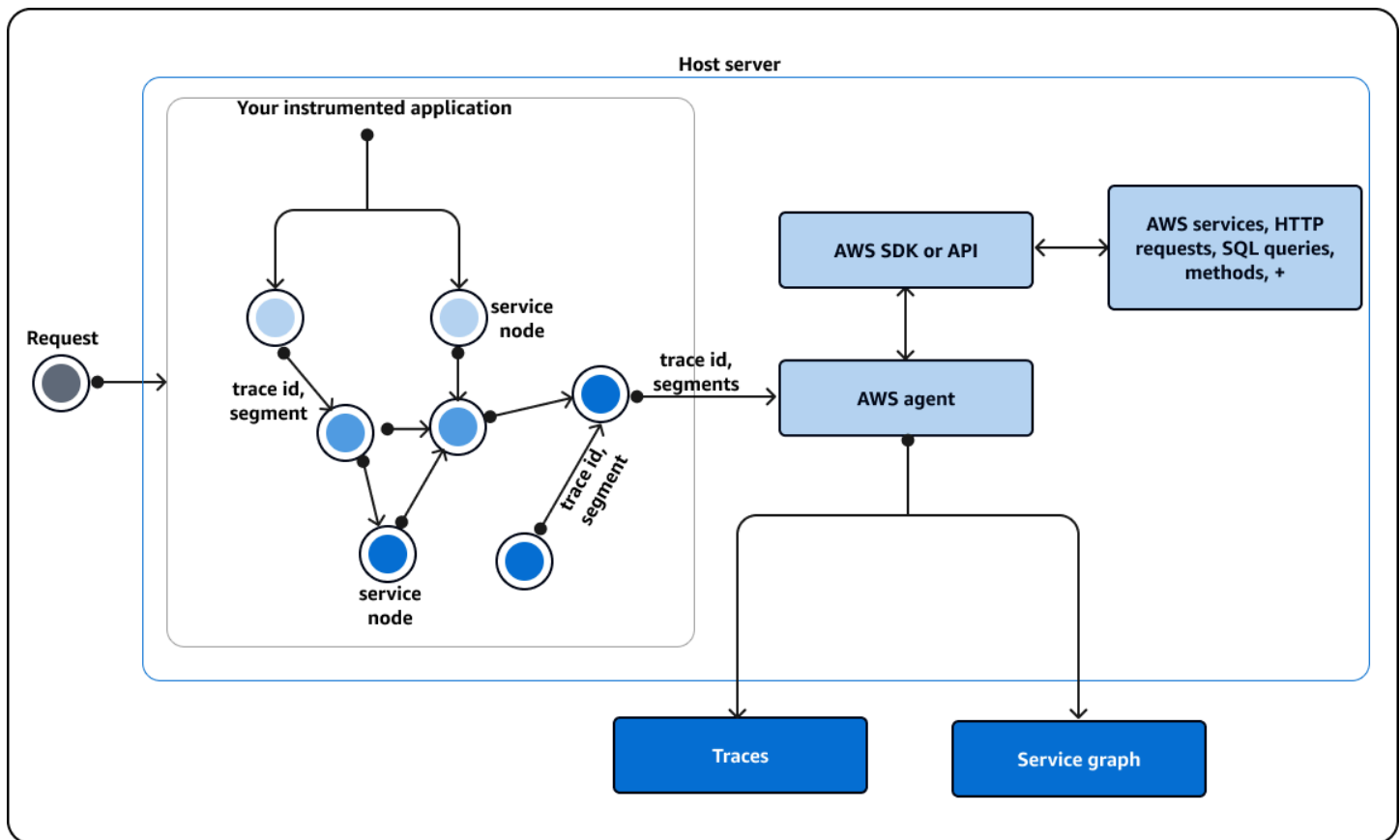
O que é o AWS X-Ray?

O AWS X-Ray é um serviço que coleta dados sobre solicitações atendidas pela aplicação e fornece ferramentas que podem ser usadas para visualizar, filtrar e obter informações sobre esses dados e identificar problemas e oportunidades de otimização. Para qualquer solicitação rastreada para a aplicação, é possível ver informações detalhadas não apenas sobre a solicitação e a resposta, mas também sobre chamadas que a aplicação faz para APIs da web, recursos, microserviços, bancos de dados da AWS subsequentes.



O AWS X-Ray recebe rastreamentos da aplicação, além dos Serviços da AWS que a aplicação usa que já estão integrados ao X-Ray. A instrumentação de uma aplicação requer o envio de dados de rastreamento para solicitações de entrada e saída e outros eventos dentro da aplicação, bem como metadados sobre cada solicitação. Muitos cenários de instrumentação exigem apenas alterações de configuração. Por exemplo, você pode instrumentar todas as solicitações HTTP de entrada e as chamadas subsequentes aos Serviços da AWS que a aplicação Java faz. Há vários SDKs, agentes e ferramentas que podem ser usados para instrumentar a aplicação para rastreamento do X-Ray. Consulte [Instrumentar sua aplicação](#) para obter mais informações.

Os Serviços da AWS que são [integrados ao X-Ray](#) podem adicionar cabeçalhos de rastreamento às solicitações de entrada, enviar dados de rastreamento ao X-Ray ou executar o daemon do X-Ray. Por exemplo, o AWS Lambda pode enviar dados de rastreamento sobre as solicitações para as funções do Lambda e executar o daemon do X-Ray em operadores para simplificar o uso do X-Ray SDK.



Em vez de enviar dados de rastreamento diretamente para o X-Ray, cada SDK de cliente envia documentos segmentados JSON a um processo do daemon que escuta o tráfego UDP. O [daemon do X-Ray](#) armazena os segmentos em buffer em uma fila e os carrega em lote no X-Ray. O daemon está disponível para Linux, Windows e macOS e está incluído nas plataformas AWS Elastic Beanstalk e AWS Lambda.

O X-Ray usa dados de rastreamento dos recursos da AWS que preparam as aplicações de nuvem para gerar um mapa de rastreamento detalhado. O mapa de rastreamento mostra o cliente, o serviço de front-end e os serviços de backend chamados pelo serviço de front-end para processar solicitações e manter dados. Use o mapa de rastreamento para identificar gargalos, picos de latência e outros problemas e melhorar o desempenho das aplicações.



Conceitos básicos do X-Ray

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Para usar o X-Ray, siga estas etapas:

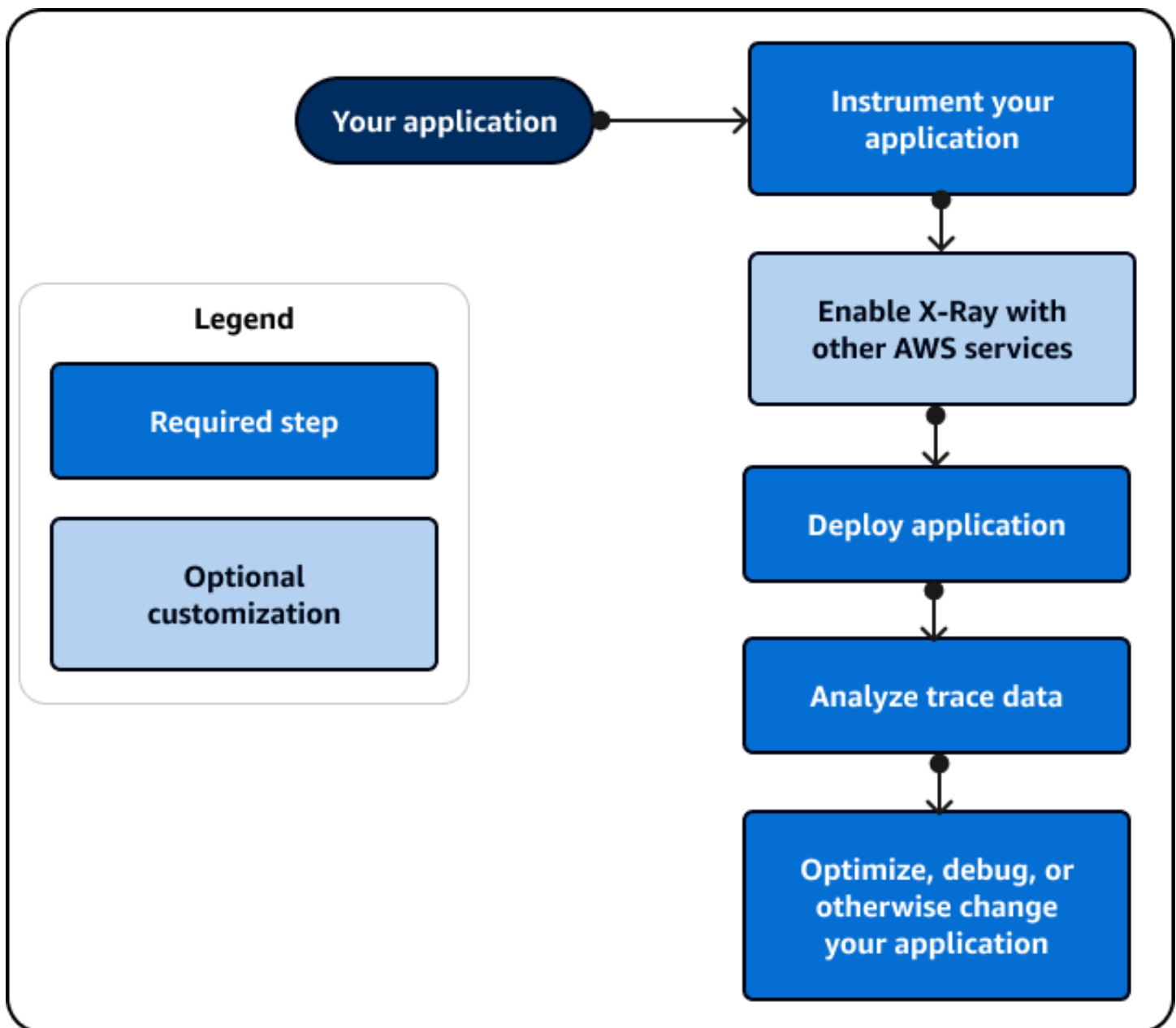
1. Instrumente seu aplicativo, o que permite que o X-Ray rastreie como ele processa uma solicitação.
 - Use o X-Ray SDKs, o X-Ray APIs ADOT ou o CloudWatch Application Signals para enviar dados de rastreamento para o X-Ray. Para obter mais informações sobre a interface a ser usada, consulte [Escolher uma interface](#).

Para obter mais informações sobre a instrumentação, consulte [Instrumentando seu aplicativo para AWS X-Ray](#).

2. (Opcional) Configure o X-Ray para funcionar com outros Serviços da AWS que se integram ao X-Ray. Você pode fazer amostras de rastreamentos e adicionar cabeçalhos a solicitações de entrada, executar um agente ou coletor e enviar dados de rastreamento automaticamente ao X-Ray. Para obter mais informações, consulte [Integrando AWS X-Ray com outros Serviços da AWS](#).
3. Implantar o aplicativo instrumentado. Conforme seu aplicativo recebe solicitações, o X-Ray SDK registra dados de rastreamento, segmento e subsegmento. Nesta etapa, talvez você também precise configurar uma política do IAM e implantar um agente ou coletor.
 - Por exemplo, scripts para implantar um aplicativo usando o SDK AWS Distro for OpenTelemetry (ADOT) e o CloudWatch agente em diferentes plataformas, consulte [Application Signals Demo Scripts](#).

- Para obter um script de exemplo para implantar o aplicativo usando o SDK e o daemon do X-Ray, consulte [AWS X-Ray aplicação de amostra](#).
4. (Opcional) Abra um console para visualizar e analisar os dados. Você pode ver uma representação gráfica de um mapa de rastreamento, mapa de serviço e muito mais para inspecionar o funcionamento do aplicativo. Use as informações gráficas no console para otimizar, depurar e entender seu aplicativo. Para obter informações sobre como escolher um console, consulte [Usar um console](#).

O diagrama a seguir mostra como começar a usar o X-Ray:



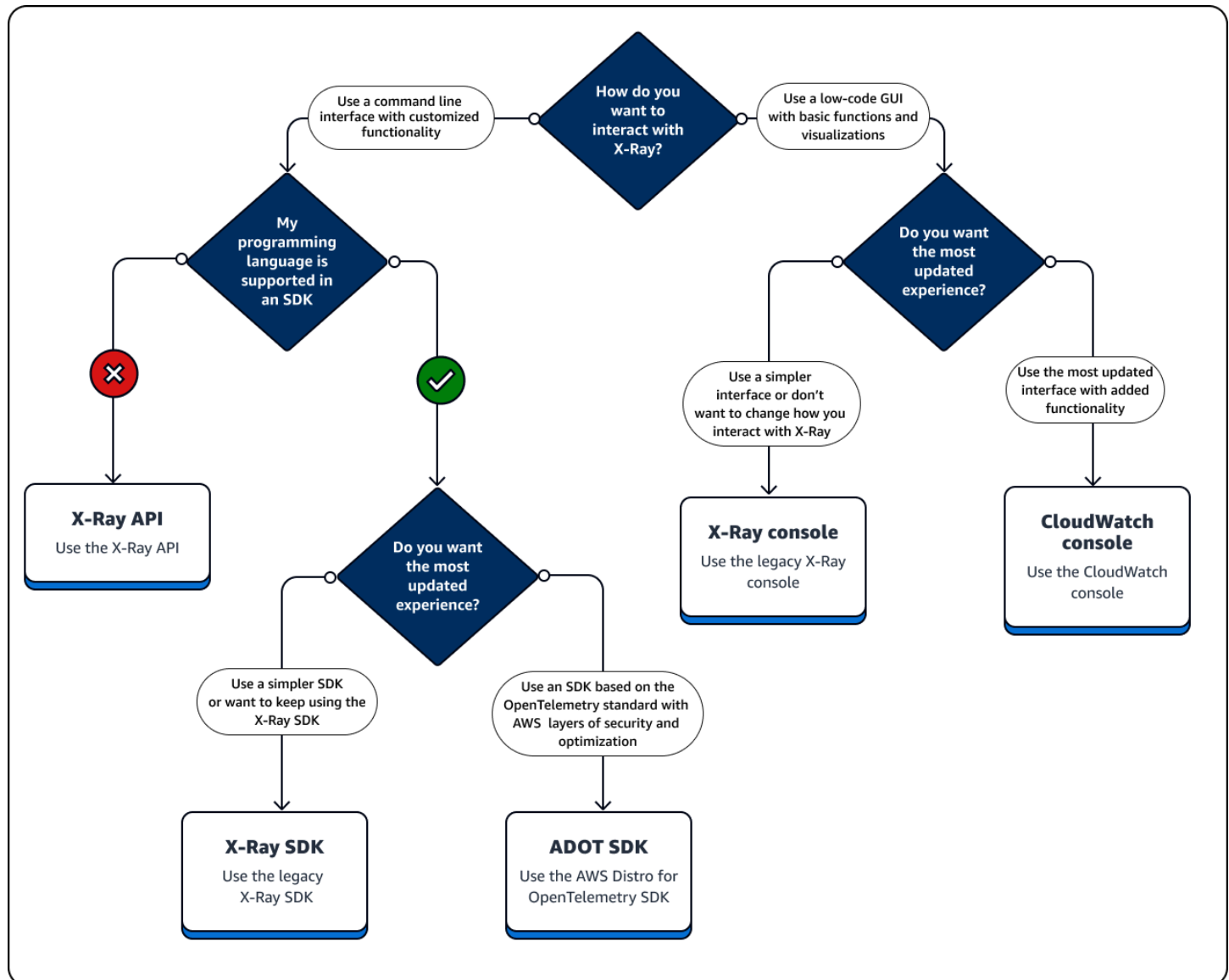
Para obter um exemplo dos dados e mapas disponíveis no console, inicie um aplicativo [de amostra](#) que já esteja instrumentada para gerar dados de rastreamento. Em poucos minutos, você pode gerar tráfego, enviar segmentos para o X-Ray e visualizar um mapa de rastreamento e serviço.

Escolher uma interface

AWS X-Ray pode fornecer informações sobre como seu aplicativo funciona e como ele interage com outros serviços e recursos. Depois de instrumentar ou configurar o aplicativo, o X-Ray coleta dados de rastreamento à medida que ela atende às solicitações. Você pode analisar esses dados de rastreamento para identificar problemas de desempenho, solucionar erros e otimizar recursos. Este guia mostra como interagir com o X-Ray com as seguintes diretrizes:

- Use um Console de gerenciamento da AWS se quiser começar rapidamente ou se puder usar visualizações pré-criadas para realizar tarefas básicas.
 - Escolha o CloudWatch console da Amazon para obter a experiência de usuário mais atualizada que contém todas as funcionalidades do console X-Ray.
 - Use o console do X-Ray se quiser uma interface mais simples ou não quiser alterar a forma como você interage com o X-Ray.
- Use um SDK se precisar de mais recursos personalizados de rastreamento, monitoramento ou registro do que um Console de gerenciamento da AWS pode fornecer.
 - Escolha o SDK do ADOT se quiser um SDK independente de fornecedor baseado no SDK do OpenTelemetry de código aberto com camadas adicionais de segurança e otimização da AWS .
 - Escolha o X-Ray SDK se quiser um SDK mais simples ou não quiser atualizar o código do aplicativo.
- Use as operações da API do X-Ray se um SDK não for compatível com a linguagem de programação do aplicativo.

O diagrama a seguir ajuda você a escolher como interagir com o X-Ray:



Explore os tipos de interface

- [Usar um SDK](#)
- [Usar um console](#)
- [Usar a API do X-Ray](#)

Usar um SDK

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Use um SDK se quiser usar uma interface de linha de comando ou precisar de mais recursos personalizados de rastreamento, monitoramento ou log do que os disponíveis em um Console de gerenciamento da AWS. Você também pode usar um AWS SDK para desenvolver programas que usem o X-Ray APIs. Você pode usar o AWS Distro for OpenTelemetry (ADOT) SDK ou o X-Ray SDK.

Ao usar um SDK, você pode adicionar personalizações ao fluxo de trabalho ao instrumentar seu aplicativo e ao configurar o coletor ou o agente. Você pode usar um SDK para realizar as seguintes tarefas, o que não é possível usando um Console de gerenciamento da AWS:

- Publicar métricas personalizadas: avalie métricas em altas resoluções de até 1 segundo, use várias dimensões para adicionar informações sobre uma métrica e agregue pontos de dados em um conjunto de estatísticas.
- Personalizar seu coletor: personalize a configuração de qualquer parte de um coletor, incluindo o receptor, o processador, o exportador e o conector.
- Personalizar sua instrumentação: personalize segmentos e subsegmentos, adicione pares de valor-chave personalizados como atributos e crie métricas personalizadas.
- Crie e atualize regras de amostragem programaticamente.

Use o ADOT SDK se quiser a flexibilidade de usar um OpenTelemetry SDK padronizado com camadas adicionais de AWS segurança e otimização. O AWS Distro for OpenTelemetry (ADOT) SDK é um pacote independente de fornecedor que permite a integração com back-ends de outros fornecedores e não prestadores de AWS serviços sem precisar reinstrumentar seu código.

Use o X-Ray SDK se você já estiver usando o X-Ray SDK, se integrar apenas com backends da AWS e não quiser alterar a maneira como interage com o X-Ray ou com o código do aplicativo.

Para obter mais informações sobre cada recurso, consulte [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#).

Usar o SDK do ADOT

O ADOT SDK é um conjunto de bibliotecas e agentes de código APIs aberto que enviam dados para serviços de back-end. ADOT é suportado por AWS, se integra a vários back-ends e agentes e fornece um grande número de bibliotecas de código aberto mantidas pela OpenTelemetry comunidade. Use o SDK do ADOT para instrumentar seu aplicativo e coletar logs, metadados, métricas e rastreamentos. Você também pode usar ADOT para monitorar serviços e definir um alarme com base em suas métricas em CloudWatch.

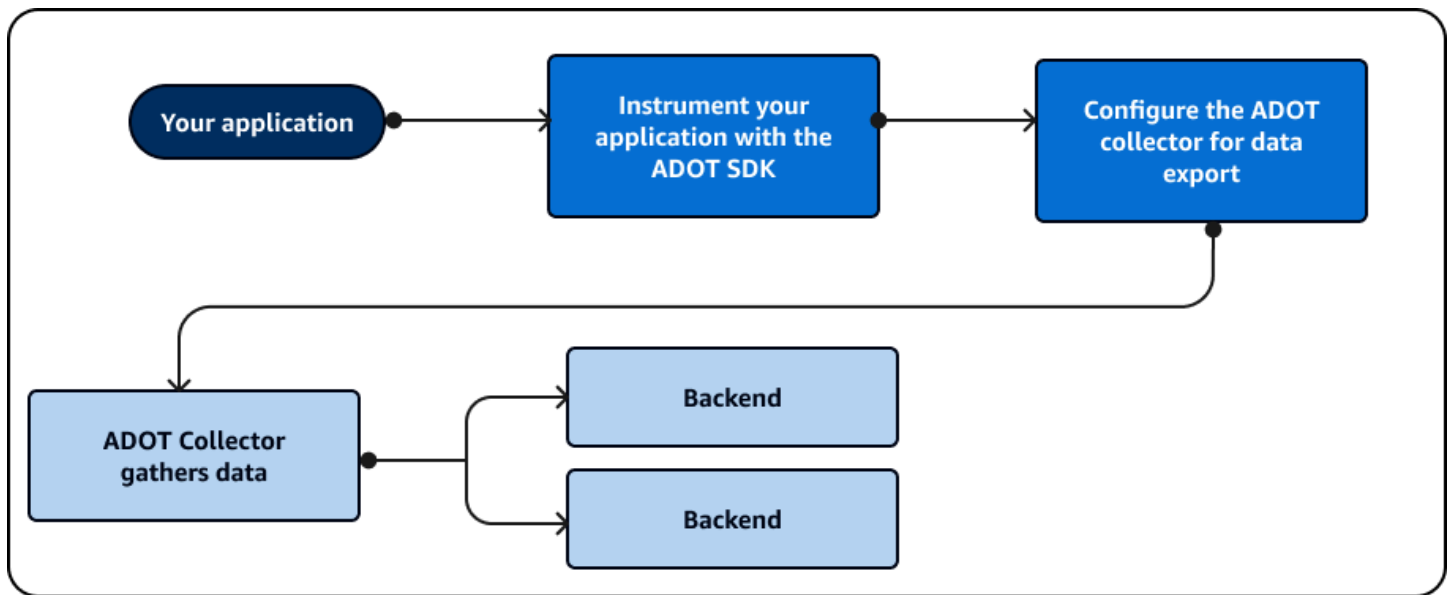
Ao usar o SDK do ADOT, você tem as seguintes opções, em combinação com um agente:

- Use o ADOT SDK com o [CloudWatch agente](#) — recomendado.
- Use o ADOT SDK com o [ADOTCollector](#) — recomendado se você quiser usar um software independente de fornecedor com AWS camadas de segurança e otimização.

Para usar o SDK do ADOT, faça o seguinte:

- Instrumente seu aplicativo usando o SDK do ADOT. Para obter mais informações, consulte a documentação de sua linguagem de programação na [documentação técnica do ADOT](#).
- Configure um coletor do ADOT para informá-lo para onde enviar os dados que ele coleta.

Depois que o ADOT coletor recebe seus dados, ele os envia para o back-end especificado na ADOT configuração. ADOT pode enviar dados para vários back-ends, inclusive para fornecedores externos AWS, conforme mostrado no diagrama a seguir:



AWS ADOT atualiza regularmente para adicionar funcionalidades e se alinhar à [OpenTelemetry](#) estrutura. Atualizações e futuros planos de desenvolvimento do ADOT fazem parte de um [roteiro](#) que está disponível para o público. O ADOT é compatível com várias linguagens de programação que incluem o seguinte:

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

Se você estiver usando Python, o ADOT pode instrumentar automaticamente seu aplicativo. Para começar a usar ADOT, consulte [Introdução e Introdução à AWS distribuição do OpenTelemetry Collector](#).

Usar o SDK do X-Ray

O X-Ray SDK é um conjunto AWS APIs e bibliotecas que enviam dados para serviços de AWS back-end. Use o X-Ray SDK para instrumentar o aplicativo e coletar dados de rastreamento. Não é possível usar o X-Ray SDK para coletar dados métricos ou de log.

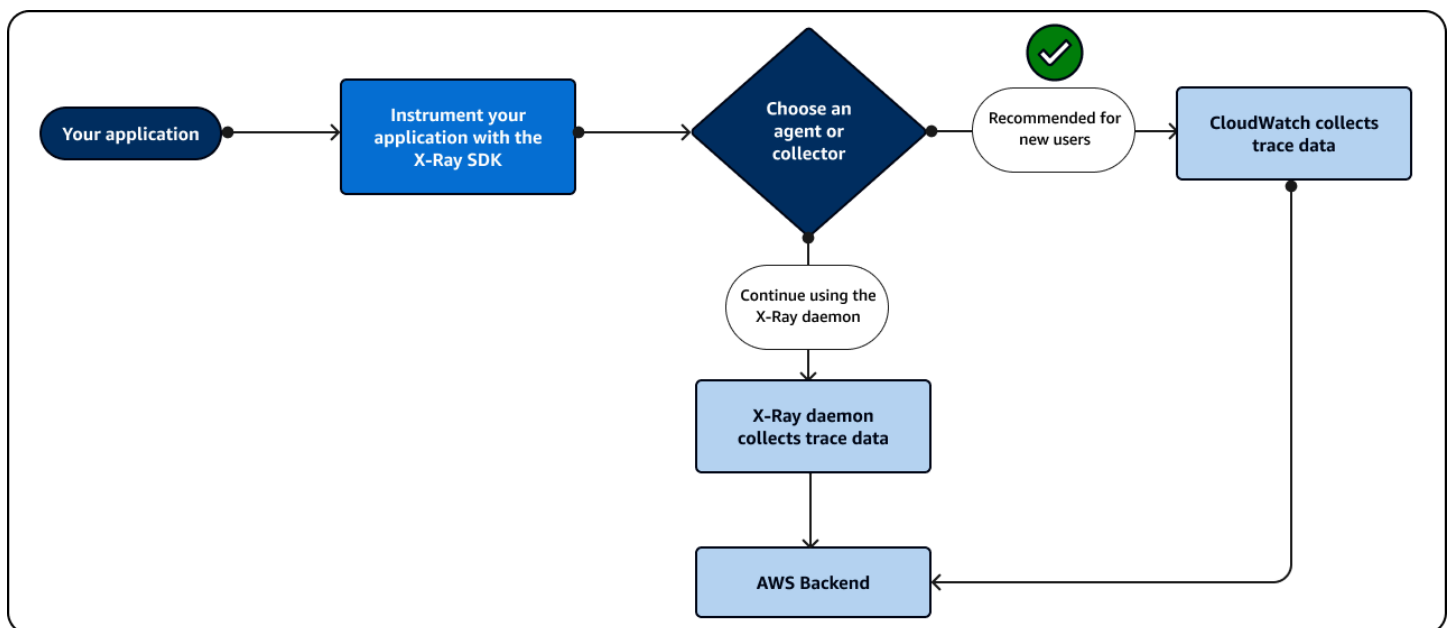
Ao usar o X-Ray SDK, você tem as seguintes opções, em combinação com um agente:

- Usar o X-Ray SDK com o [AWS X-Ray daemon](#): use isso se não quiser atualizar o código do aplicativo.
- Use o X-Ray SDK com o CloudWatch agente — (recomendado) O CloudWatch agente é compatível com o X-Ray SDK.

Para usar o X-Ray SDK, faça o seguinte:

- Instrumente seu aplicativo usando o X-Ray SDK.
- Configure um coletor para informá-lo para onde enviar os dados que ele coleta. Você pode usar o CloudWatch agente ou o daemon X-Ray para coletar suas informações de rastreamento.

Depois que o coletor ou agente recebe seus dados, ele os envia para um AWS back-end que você especifica na configuração do agente. O X-Ray SDK só pode enviar dados para um backend da AWS , conforme mostrado no diagrama a seguir:



Se estiver usando o Java, você pode usar o X-Ray SDK para instrumentar automaticamente seu aplicativo. Para começar a usar o X-Ray SDK, consulte as bibliotecas associadas às seguintes linguagens de programação:

- [Go](#)
- [Java](#)

- [Node.js](#)
- [Python](#)
- [.NET](#)
- [Ruby](#)

Usar um console

Use um console se quiser uma interface gráfica do usuário (GUI) que exija o mínimo de codificação. Os usuários iniciantes no X-Ray podem começar rapidamente a usar visualizações pré-criadas e a executar tarefas básicas. Você pode fazer o seguinte diretamente no console:

- Habilite o X-Ray.
- Ver resumos gerais do desempenho da aplicação.
- Verificar a integridade de suas aplicações.
- Identificar erros gerais.
- Ver resumos básicos de rastreamento.

Você pode usar o console do Amazon CloudWatch em <https://console.aws.amazon.com/cloudwatch/> ou o console do X-Ray em <https://console.aws.amazon.com/xray/home> para interagir com o X-Ray.

Usar o console do Amazon CloudWatch

O console do CloudWatch inclui a nova funcionalidade X-Ray que foi reprojetaada a partir do console do X-Ray para facilitar o uso. Se você usar o console do CloudWatch, poderá visualizar os logs e métricas do CloudWatch juntamente com os dados de rastreamento do X-Ray. Use o console do CloudWatch para visualizar e analisar dados, incluindo os seguintes:

- Rastreamentos do X-Ray: visualize, analise e filtre os rastreamentos associados à sua aplicação à medida que ela atende a uma solicitação. Use esses rastreamentos para encontrar altas latências, depurar erros e otimizar o fluxo de trabalho da sua aplicação. Visualize um mapa de rastreamento e um mapa de serviços para ver representações visuais do fluxo de trabalho da sua aplicação.
- Logs: visualize, analise e filtre os logs que sua aplicação produz. Use logs para solucionar erros e configurar o monitoramento com base em valores de log específicos.
- Métricas: meça e monitore o desempenho da sua aplicação usando métricas que seus recursos emitem ou crie suas próprias métricas. Visualize essas métricas em gráficos e tabelas.

- Monitoramento de redes e infraestrutura: monitore as principais redes em busca de interrupções e a integridade e o desempenho de sua infraestrutura, incluindo aplicações em contêineres, outros serviços da AWS e clientes.
- Todas as funcionalidades do console do X-Ray listadas na seção Usar o console do X-Ray a seguir.

Para obter mais informações sobre o console do CloudWatch, consulte [Conceitos básicos do Amazon CloudWatch](#).

Faça login no console do Amazon CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.

Usar o console do X-Ray

O console do X-Ray oferece rastreamento distribuído para solicitações de aplicações. Use o console do X-Ray se quiser uma experiência de console mais simples ou não quiser atualizar o código do aplicativo. A AWS não está mais desenvolvendo o console do X-Ray. O console do X-Ray contém os seguintes recursos para aplicações instrumentadas:

- [Insights](#): detecte automaticamente anomalias no desempenho da sua aplicação e encontre as causas subjacentes. Os Insights estão incluídos no console do CloudWatch em Insights. Para obter mais informações, consulte Usar insights do X-Ray no [Usar o console do X-Ray](#).
- Mapa de serviços: visualize uma estrutura gráfica da sua aplicação e conexões com clientes, recursos, serviços e dependências.
- Traços: veja uma visão geral dos rastreamentos que são gerados pela sua aplicação quando ela atende a uma solicitação. Use dados de rastreamento para entender o desempenho da sua aplicação em relação às métricas básicas, incluindo resposta HTTP e tempo de resposta.
- Análise: interprete, explore e analise dados de rastreamento usando gráficos para distribuição do tempo de resposta.
- Configuração: crie rastreamentos personalizados para alterar as configurações padrão para o seguinte:
 - Amostragem: crie uma regra que defina a frequência da amostragem da aplicação para obter informações de rastreamento. Para obter mais informações, consulte Configurar regras de amostragem no [Usar o console do X-Ray](#).
 - [Criptografia](#): criptografe dados em repouso usando uma chave que você pode auditar ou desativar usando o AWS Key Management Service.

- Grupos: use uma expressão de filtro para definir um grupo de rastreamentos com um recurso comum, como o nome de um URL ou um tempo de resposta. Para obter mais informações, consulte [Configurar grupos](#).

Faça login no console do X-Ray em <https://console.aws.amazon.com/xray/home>.

Explorar o console do X-Ray

Use o console do X-Ray para visualizar um mapa dos serviços e rastreamentos associados às solicitações atendidas pelas aplicações e para configurar grupos e regras de amostragem que afetam a forma como os rastreamentos são enviados ao X-Ray.

Note

O mapa de serviços do X-Ray e o mapa do CloudWatch ServiceLens foram combinados no mapa de rastreamento do X-Ray no console do Amazon CloudWatch. Abra o [console do CloudWatch](#) e escolha Mapa de rastreamento em Rastreamentos do X-Ray no painel de navegação esquerdo.

O CloudWatch agora inclui o [Application Signals](#), que pode descobrir e monitorar serviços das suas aplicações, clientes, canários do Synthetics e dependências de serviços. Use o Application Signals para ver uma lista ou um mapa visual dos seus serviços, visualizar métricas de integridade com base nos seus objetivos de nível de serviço (SLOs) e fazer uma busca profunda para ver rastreamentos do X-Ray correlacionados para uma solução de problemas mais detalhada.

A página principal do console do X-Ray é o Mapa de rastreamento, que é uma representação visual do gráfico de serviço JSON que o X-Ray gera com base nos dados de rastreamento gerados pelas aplicações. O mapa consiste em nós de serviço para cada aplicativo na sua conta que atende solicitações, nós de cliente upstream que representam as origens das solicitações e nós de serviço downstream que representam serviços da Web e recursos usados por um aplicativo ao processar uma solicitação. Há páginas adicionais para visualizar rastreamentos e detalhes de rastreamento e configurar grupos e regras de amostragem.

Veja a experiência de console do X-Ray e compare com o console do CloudWatch nas seções a seguir.

Explore os consoles do X-Ray e do CloudWatch

- [Usar o mapa de rastreamento do X-Ray](#)
- [Visualizar rastreamentos e detalhes do rastreamento](#)
- [Usar expressões de filtro](#)
- [Rastreamento entre contas](#)
- [Rastrear aplicações orientadas a eventos](#)
- [Usar histogramas de latência](#)
- [Usar os insights do X-Ray](#)
- [Interagir com o console do Analytics](#)
- [Configurar grupos](#)
- [Configurar regras de amostragem](#)
- [Configurar amostragem adaptável](#)
- [Vinculação direta do console](#)

Usar o mapa de rastreamento do X-Ray

Visualize o mapa de rastreamento do X-Ray para identificar os serviços nos quais estão ocorrendo erros, conexões com alta latência ou rastreamentos de solicitações com erros.

Note

O CloudWatch agora inclui o [Application Signals](#), que pode descobrir e monitorar serviços das suas aplicações, clientes, canários do Synthetics e dependências de serviços. Use o Application Signals para ver uma lista ou um mapa visual dos seus serviços, visualizar métricas de integridade com base nos seus objetivos de nível de serviço (SLOs) e fazer uma busca profunda para ver rastreamentos do X-Ray correlacionados para uma solução de problemas mais detalhada.

O mapa de serviços do X-Ray e o mapa do CloudWatch ServiceLens foram combinados no mapa de rastreamento do X-Ray no console do Amazon CloudWatch. Abra o [console do CloudWatch](#) e escolha Mapa de rastreamento em Rastreamentos do X-Ray no painel de navegação esquerdo.

Exibir o mapa de rastreamento

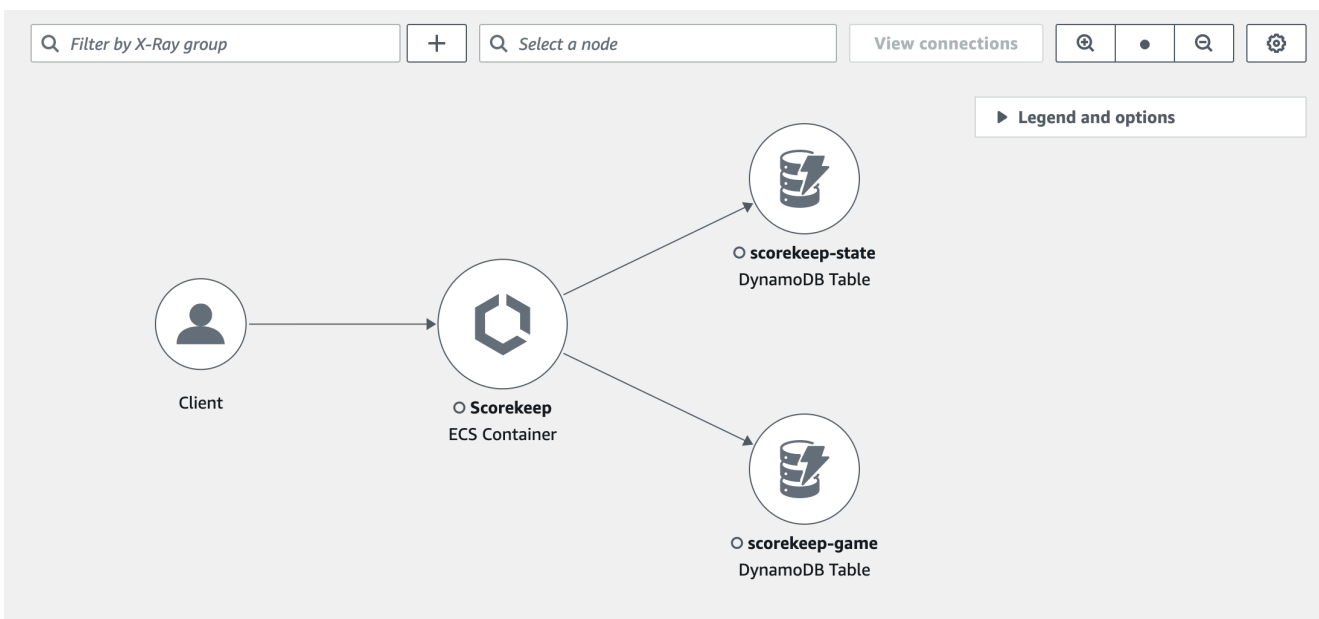
O mapa de rastreamento é uma representação visual dos dados de rastreamento gerados pelas aplicações. Ele mostra nós de serviço que atendem a solicitações, nós de cliente precedentes que representam as origens das solicitações e nós de serviço subsequentes que representam serviços da web e recursos usados por uma aplicação ao processar uma solicitação.

O mapa de rastreamento exibe uma visão conectada dos rastreamentos em aplicações orientadas a eventos que usam o Amazon SQS e o Lambda. Para obter mais informações, consulte [Rastrear aplicações orientadas a eventos](#). O mapa de rastreamento também é compatível com [rastreamento entre contas](#), exibindo nós de várias contas em um único mapa.

CloudWatch console

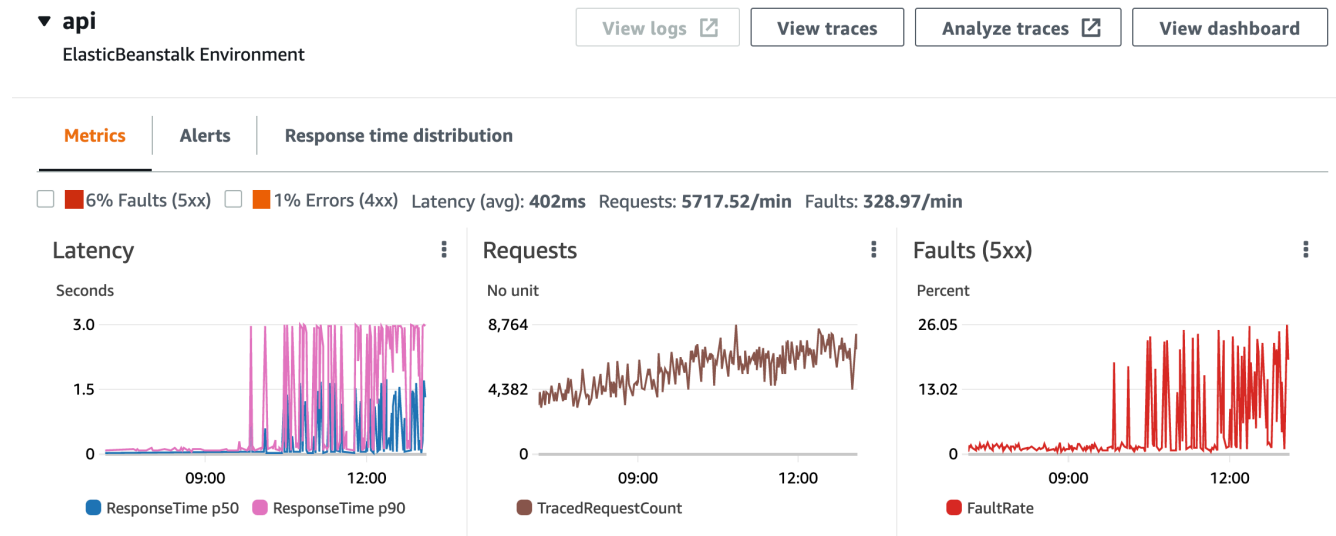
Como visualizar o mapa de rastreamento no console do CloudWatch

1. [Abra o console do CloudWatch](#). Escolha Mapa de rastreamento na seção Rastreamentos do X-Ray no painel de navegação esquerdo.



2. Escolha um nó de serviço para visualizar solicitações desse nó ou uma borda entre dois nós para visualizar solicitações que percorreram essa conexão.
3. Informações adicionais são exibidas abaixo do mapa de rastreamento, incluindo guias para métricas, alertas e distribuição do tempo de resposta. Na guia Métricas, selecione um intervalo em cada gráfico para ver mais detalhes ou escolha as opções Falhas ou Erros para

filtrar os rastreamentos. Na guia Distribuição do tempo de resposta, selecione um intervalo no gráfico para filtrar os rastreamentos por tempo de resposta.



4. Para visualizar os rastreamentos, escolha Visualizar rastreamentos ou, se tiver sido aplicado um filtro, selecione Exibir rastreamentos filtrados.
5. Escolha Visualizar logs para ver os logs do CloudWatch associados ao nó selecionado. Nem todos os nós do mapa de rastreamento oferecem suporte à visualização de logs. Consulte [Solução de problemas de logs do CloudWatch](#) para obter mais informações.

O mapa de rastreamento indica problemas em cada nó por meio de cores:

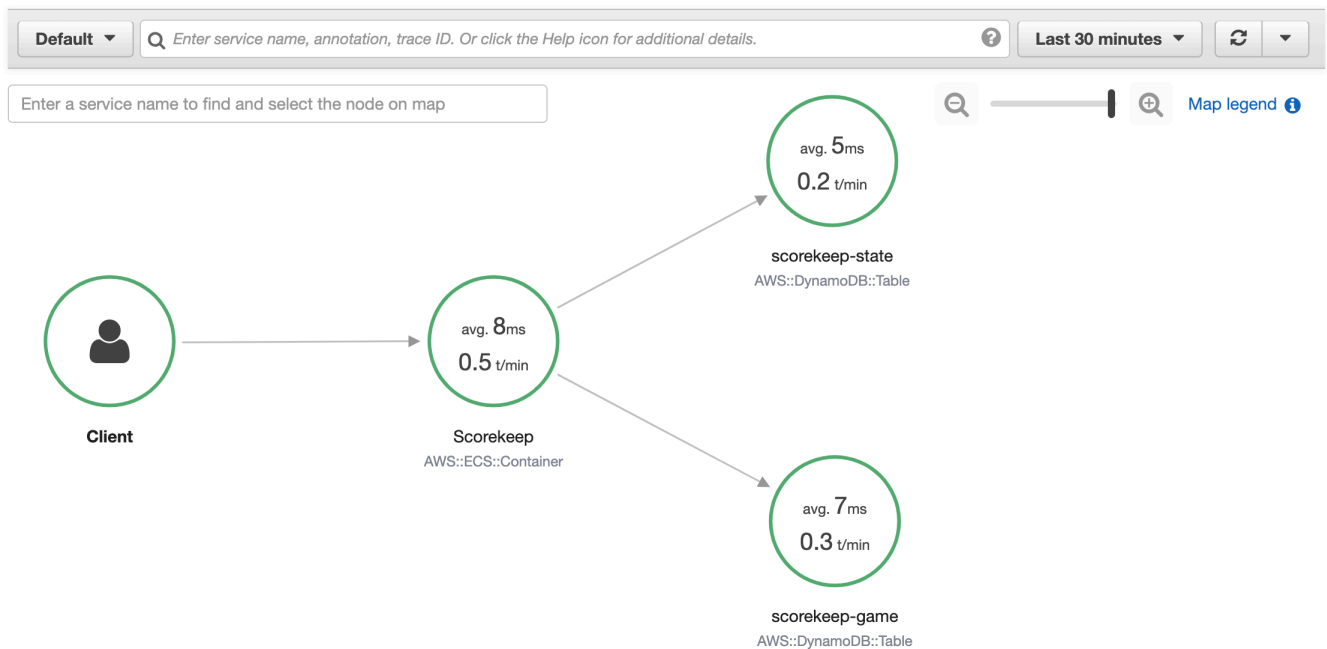
- Vermelho para falhas do servidor (erros da série 500)
- Amarelo para erros de clientes (erros da série 400)
- Roxo para erros de controle de utilização (429, muitas solicitações)

Se o mapa de rastreamento for grande, use os controles na tela ou o mouse para aumentar e diminuir o zoom e mover o mapa.

X-Ray console

Para visualizar o mapa de serviço

1. Abra o [console do X-Ray](#). O mapa de serviço é exibido por padrão. Você também pode escolher Mapa de serviço no painel de navegação esquerdo.



2. Escolha um nó de serviço para visualizar solicitações desse nó ou uma borda entre dois nós para visualizar solicitações que percorreram essa conexão.
3. Use o [histograma](#) de distribuição de resposta para filtrar rastreamentos por duração e selecione códigos de status cujos rastreamentos você deseja visualizar. Em seguida, escolha Visualizar rastreamentos para abrir a lista de rastreamentos com a expressão de filtro aplicada.

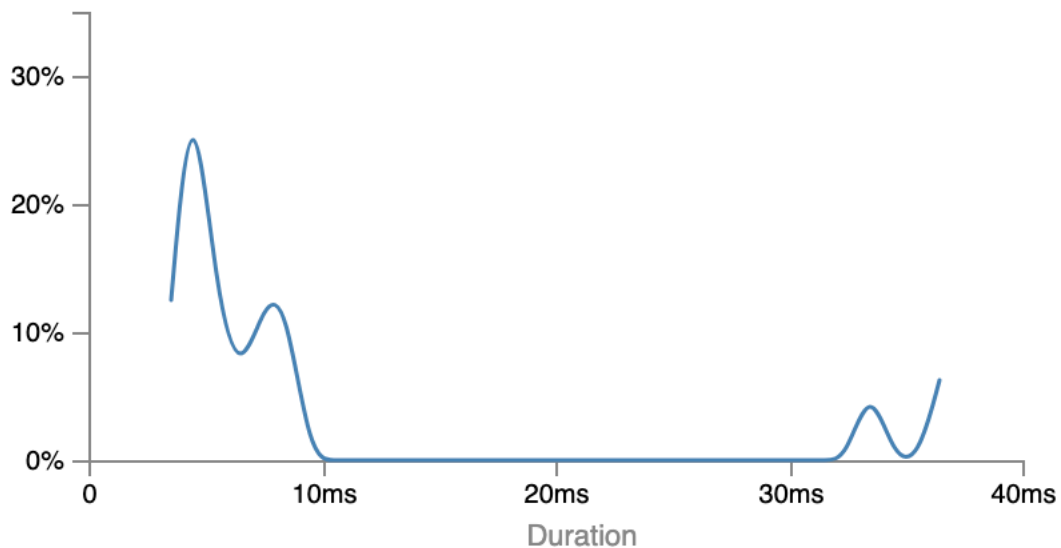
Service details ?

Name: Scorekeep

Type: AWS::ECS::Container

Response distribution

Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.



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 >

O mapa de serviço indica a integridade de cada nó atribuindo cores a ele com base no índice de chamadas bem-sucedidas em relação a erros e falhas:

- Verde para chamadas bem-sucedidas
- Vermelho para falhas do servidor (erros da série 500)
- Amarelo para erros de clientes (erros da série 400)
- Roxo para erros de controle de utilização (429, muitas solicitações)

Se o mapa de serviço for grande, use os controles na tela ou o mouse para ampliar e reduzir e mover o mapa.

Note

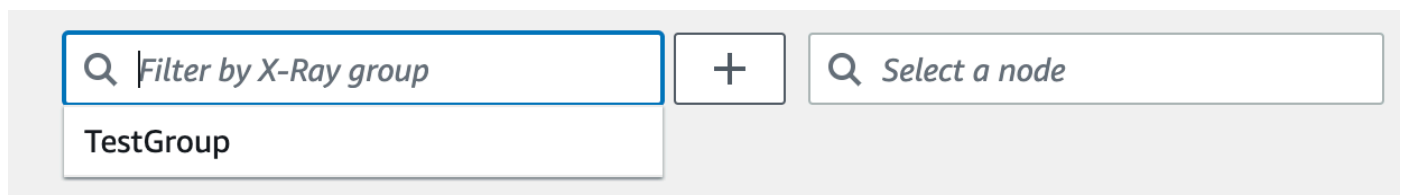
O mapa de rastreamento do X-Ray pode exibir até dez mil nós. Em situações raras em que o número total de nós de serviço excede esse limite, você pode receber um erro e não conseguir exibir um mapa de rastreamento completo no console.

Filtrar o mapa de rastreamento por grupo

Ao usar uma [expressão de filtro](#), é possível definir critérios para incluir rastreamentos em um grupo. Use as etapas a seguir para exibir esse grupo específico no mapa de rastreamento.

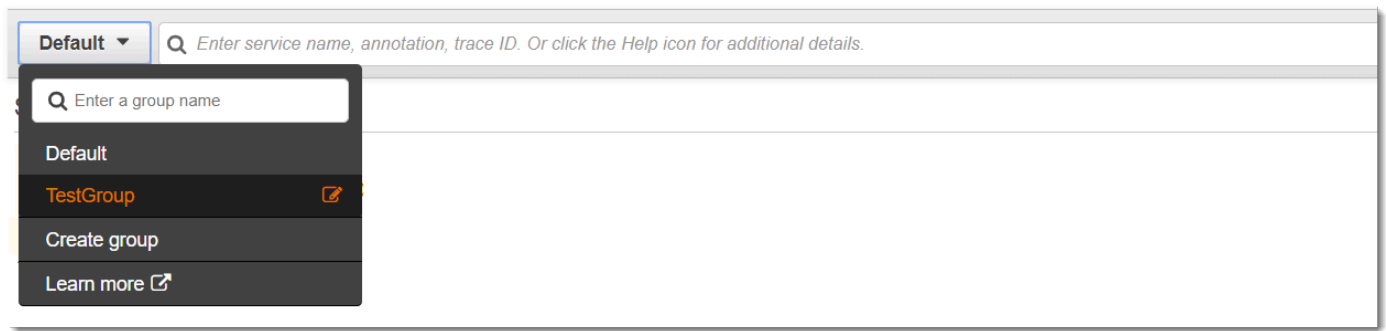
CloudWatch console

Escolha um nome de grupo no filtro de grupos no canto superior esquerdo do mapa de rastreamento.



X-Ray console

Escolha um nome de grupo no menu suspenso à esquerda da barra de pesquisa.



O mapa de serviço agora será filtrado para exibir rastreamentos que correspondam à expressão de filtro do grupo selecionado.

Legenda e opções do mapa de rastreamento

O mapa de rastreamento inclui uma legenda e várias opções para personalizar a exibição do mapa.

CloudWatch console

Escolha o menu suspenso Legenda e opções no canto superior direito do mapa. Escolha o que é exibido nos nós, como:

- Métricas: exibem o tempo médio de resposta e o número de rastreamentos enviados por minuto durante o intervalo de tempo escolhido.
- Nós: exibem o ícone do serviço em cada nó.

Escolha configurações adicionais do mapa no painel Preferências, que pode ser acessado por meio do ícone de engrenagem no canto superior direito do mapa. Essas configurações exigem a seleção da métrica a ser usada para determinar o tamanho de cada nó e dos canários que devem ser exibidos no mapa.

X-Ray console

Exiba a legenda do mapa de serviço escolhendo o link Legenda do mapa no canto superior direito do mapa. As opções do mapa de serviço podem ser escolhidas no canto inferior direito do mapa de rastreamento, como:

- Ícones de serviço: alternam o que é exibido em cada nó, exibindo o ícone do serviço ou o tempo médio de resposta e o número de rastreamentos enviados por minuto durante o intervalo de tempo escolhido.

- Dimensionamento de nó: Nenhum define todos os nós com o mesmo tamanho.
- Dimensionamento de nó: Integridade dimensiona os nós pelo número de solicitações afetadas, incluindo erros, falhas ou solicitações com controle de utilização.
- Dimensionamento de nó: Tráfego dimensiona os nós pelo número total de solicitações.

Visualizar rastreamentos e detalhes do rastreamento

Use a lista Rastreamentos no console do X-Ray para encontrar rastreamentos por URL, código de resposta ou outros dados do resumo de rastreamentos. Depois de selecionar um rastreamento na lista de rastreamentos, a página Detalhes de rastreamento exibe um mapa dos nós de serviço associados ao rastreamento selecionado e uma linha do tempo dos segmentos de rastreamento.

Visualizar os rastreamentos

CloudWatch console

Como exibir métricas no console do CloudWatch

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. No painel de navegação esquerdo, escolha Rastreamentos do X-Ray e, em seguida, selecione Rastreamentos. Você pode filtrar por grupo ou inserir uma [expressão de filtro](#). Isso filtra os rastreamentos que são exibidos na seção Rastreamentos na parte inferior da página.

Como alternativa, os rastreamentos podem ser procurados no mapa de serviço para navegar até um nó de serviço específico e visualizar os rastreamentos. Isso abre a página Rastreamentos com uma consulta já aplicada.

3. Refine sua consulta na seção Refinadores de consulta. Para filtrar rastreamentos por um atributo comum, escolha uma opção na seta para baixo ao lado de Refinar consulta por. As opções incluem o seguinte:
 - Nó: filtre rastreamentos por nó de serviço.
 - ARN do recurso: filtra os rastreamentos por um recurso associado a um rastreamento. Exemplos desses recursos incluem uma instância do Amazon Elastic Compute Cloud (Amazon EC2), uma função do AWS Lambda ou uma tabela do Amazon DynamoDB.
 - Usuário: filtre rastreamentos com um ID de usuário.
 - Mensagem de causa raiz do erro: filtra os rastreamentos pela causa raiz do erro.

- URL: filtre os rastreamentos por um caminho de URL usado pela aplicação.
- Código de status HTTP: filtre os rastreamentos pelo código de status HTTP retornado pela aplicação. Você pode especificar um código de resposta personalizado ou selecionar uma das seguintes opções:
 - 200: a solicitação foi concluída com êxito.
 - 401: a solicitação não tinha credenciais de autenticação válidas.
 - 403: a solicitação não tinha permissões válidas.
 - 404: o servidor não conseguiu encontrar o recurso solicitado.
 - 500: o servidor encontrou uma condição inesperada e gerou um erro interno.

Escolha uma ou mais entradas e selecione Adicionar à consulta para adicionar à expressão de filtro na parte superior da página.

4. Para localizar um único rastreamento, insira um [ID de rastreamento](#) diretamente no campo de consulta. Você pode usar o formato X-Ray ou o formato World Wide Web Consortium (W3C). Por exemplo, um rastreamento criado usando o [AWS Distro for OpenTelemetry](#) está no formato W3C.

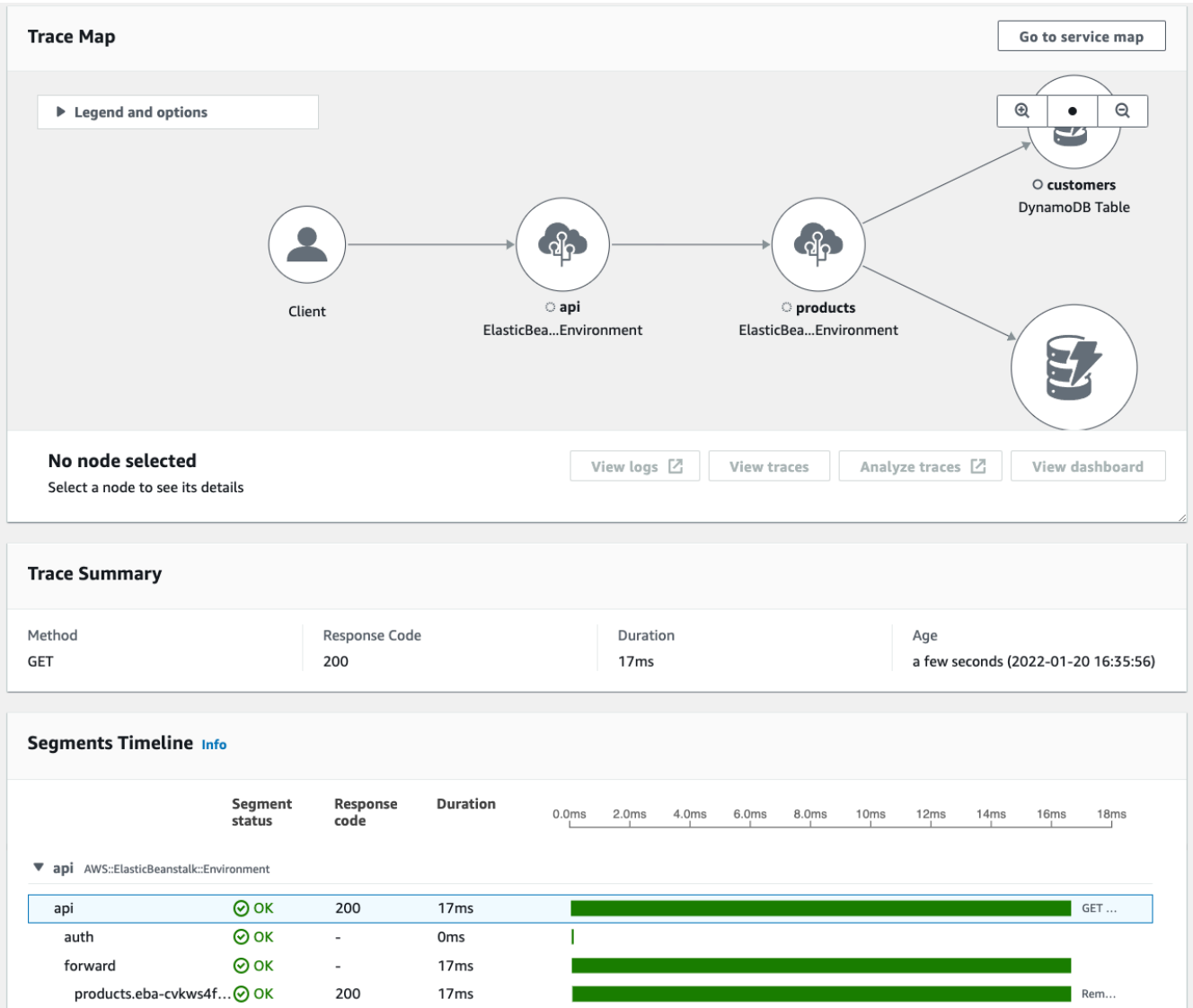
 Note

Ao consultar os rastreamentos criados com um ID de rastreamento no formato W3C, o console exibe o rastreamento correspondente no formato X-Ray. Por exemplo, se você consultar 4efaaaf4d1e8720b39541901950019ee5 no formato W3C, o console exibirá o equivalente ao X-Ray: 1-4efaaaf4d-1e8720b39541901950019ee5.

5. Escolha Executar consulta a qualquer momento para exibir uma lista de rastreamentos correspondentes na seção Rastreamentos na parte inferior da página.
6. Para exibir a página Detalhes do rastreamento para um único rastreamento, selecione o respectivo ID na lista.

A imagem a seguir mostra um Mapa de rastreamento contendo nós de serviço associados ao rastreamento e bordas entre os nós que representam o caminho percorrido pelos segmentos que compõem o rastreamento. Um Resumo do rastreamento acompanha o Mapa do rastreamento. O resumo contém informações sobre uma operação GET de amostra, seu Código de resposta, a Duração que o rastreamento levou para ser executado e a Idade da

solicitação. A Linha do tempo dos segmentos segue o Resumo do rastreamento, que mostra a duração dos segmentos e subsegmentos de rastreamento.



Se você tiver uma aplicação orientada a eventos que usa o Amazon SQS e o Lambda, será possível ver uma visualização conectada dos rastreamentos de cada solicitação no Mapa de rastreamento. No mapa, os rastreamentos dos produtores de mensagens são vinculados aos rastros dos consumidores do AWS Lambda e são exibidos como uma borda tracejada. Para obter mais informações sobre aplicações orientadas por eventos, consulte [Rastrear aplicações orientadas a eventos](#).

As páginas Rastreamentos e Detalhes de rastreamento também permitem [rastreamento entre contas](#), que pode listar rastreamentos de várias contas na lista de rastreamento e em um único mapa de rastreamento.

X-Ray console

Como visualizar rastreamentos no console do X-Ray

1. Abra a página [Rastreamentos](#) no console do X-Ray. O painel Visão geral do rastreamento mostra uma lista de rastreamentos que são agrupados por recursos comuns, incluindo Causas raiz do erro, ResourceARN e Instancelid.
2. Para selecionar um atributo comum para visualizar um conjunto agrupado de rastreamentos, expanda a seta para baixo ao lado de Agrupar por. A ilustração a seguir mostra uma visão geral dos rastreamentos agrupados por URL para o [AWS X-Ray aplicação de amostra](#) e uma lista dos rastreamentos associados.

Trace overview

Group by: URL

URL	Avg response time	% of Traces	Response
http://scorekeep.elasticbeanstalk.com/api/user	391 ms	4.76%	1 OK, 0 Throttled, 0 Errors, 0 Faults
http://scorekeep.elasticbeanstalk.com/api/session/8N63LUQ6	33.0 ms	4.76%	1 OK, 0 Throttled, 0 Errors, 0 Faults
http://scorekeep.elasticbeanstalk.com/api/session	90.5 ms	9.52%	2 OK, 0 Throttled, 0 Errors, 0 Faults

Trace list (21)

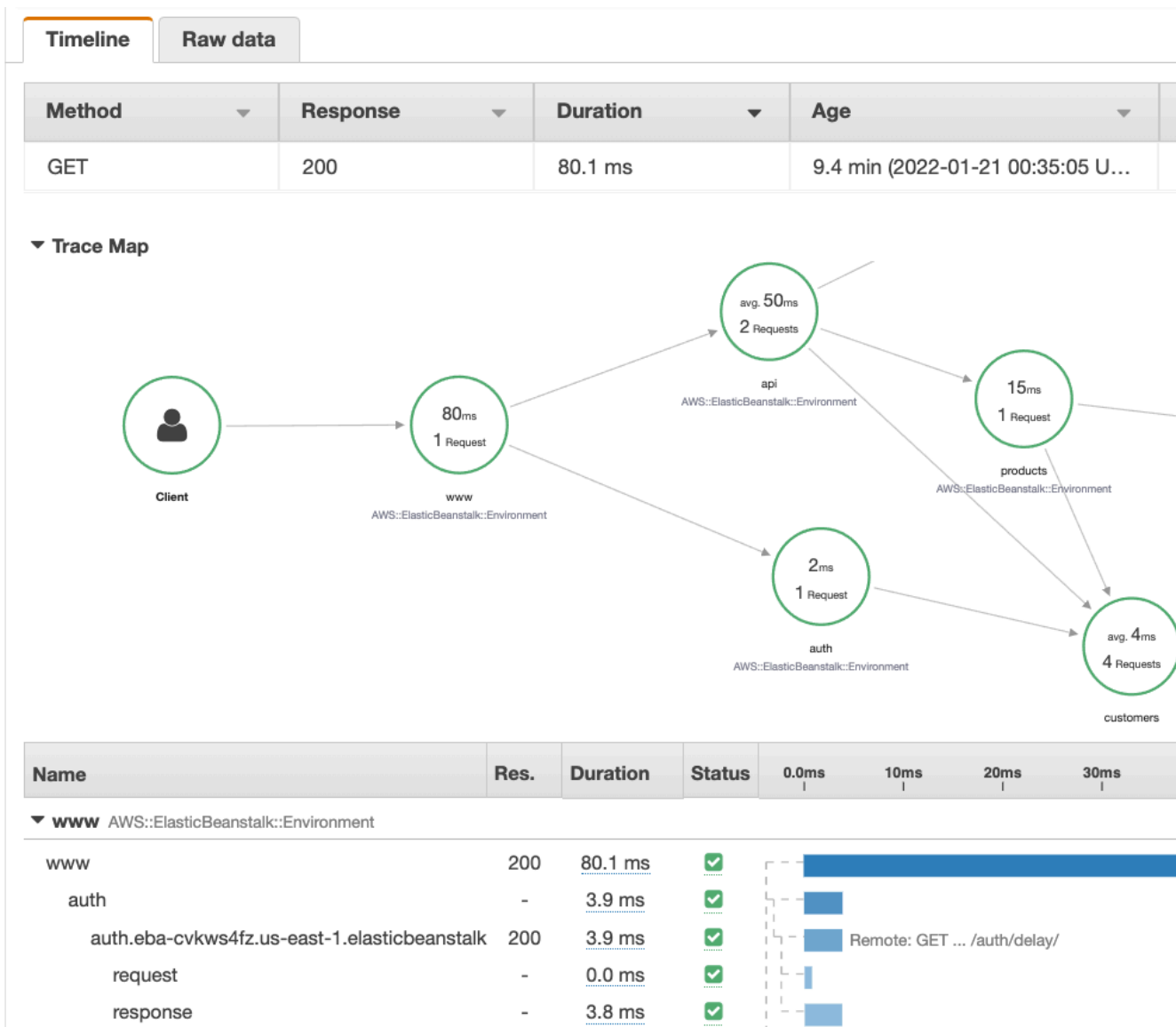
ID	Age	Method	Response	Response time	URL	Annotations
...f5f2df73	5.0 min	POST	200	391 ms	http://scorekeep.elasticbeanstalk.com/api/user	0
...cfe39980	5.0 min	PUT	200	33.0 ms	http://scorekeep.elasticbeanstalk.com/api/session/8N63LUQ6	0
...dd653e4c	5.0 min	POST	200	19.0 ms	http://scorekeep.elasticbeanstalk.com/api/session	0
...4765fec8	5.0 min	GET	200	162 ms	http://scorekeep.elasticbeanstalk.com/api/session	0
...84eeef29	4.7 min	POST	200	95.0 ms	http://scorekeep.elasticbeanstalk.com/api/move/8N63LUQ6/2N56AC7L/PPMPBLJB	1
...3ab33fdb	4.8 min	POST	200	95.0 ms	http://scorekeep.elasticbeanstalk.com/api/move/8N63LUQ6/2N56AC7L/PPMPBLJB	1
...237e0705	4.8 min	POST	200	295 ms	http://scorekeep.elasticbeanstalk.com/api/move/8N63LUQ6/2N56AC7L/PPMPBLJB	1
...86782227	4.9 min	POST	200	25.0 ms	http://scorekeep.elasticbeanstalk.com/api/game/8N63LUQ6/2N56AC7L/users	1
...fd82cc32	4.9 min	PUT	200	121 ms	http://scorekeep.elasticbeanstalk.com/api/game/8N63LUQ6/2N56AC7L/rules/TicTacToe	1
...7ca2e05f	1.4 min	GET	200	14.0 ms	http://scorekeep.elasticbeanstalk.com/api/game/8N63LUQ6/2N56AC7L	0
...062ccac5	1.7 min	GET	200	12.0 ms	http://scorekeep.elasticbeanstalk.com/api/game/8N63LUQ6/2N56AC7L	0
...dc0ebe3c	1.9 min	GET	200	9.0 ms	http://scorekeep.elasticbeanstalk.com/api/game/8N63LUQ6/2N56AC7L	0
...524637dc	4.9 min	PUT	200	69.0 ms	http://scorekeep.elasticbeanstalk.com/api/game/8N63LUQ6/2N56AC7L	1
...fdf5bb67	4.9 min	POST	200	81.0 ms	http://scorekeep.elasticbeanstalk.com/api/game/8N63LUQ6	1

3. Escolha o ID de um rastreamento para visualizá-lo na Lista de rastreamento. Você também pode escolher Mapa de serviços no painel de navegação para visualizar os rastreamentos de um nó de serviço específico. Em seguida, você pode visualizar os rastreamentos associados a esse nó.

A guia Linha do tempo mostra o fluxo de solicitações para o rastreamento e inclui o seguinte:

- Um mapa do caminho para cada segmento no rastreamento.
- Quanto tempo levou para o segmento alcançar um nó no mapa de rastreamento.
- Quantas solicitações foram feitas ao nó no mapa de rastreamento.

A ilustração a seguir mostra um exemplo de Mapa de rastreamento associado a uma solicitação GET feita a uma aplicação de amostra. As setas mostram o caminho que cada segmento percorreu para concluir a solicitação. Os nós de serviço mostram o número de solicitações feitas durante a solicitação GET.



Para obter mais informações sobre a guia Linha do tempo, consulte a seção Explorar a linha do tempo do rastreamento a seguir.

A guia Dados brutos mostra informações sobre o rastreamento e os segmentos e subsegmentos que compõem o rastreamento, no formato JSON. Essas informações podem incluir o seguinte:

- Carimbos de data/hora
- IDs exclusivos
- Recursos associados ao segmento ou subsegmento
- A fonte ou origem do segmento ou subsegmento
- Informações adicionais sobre a solicitação à sua aplicação, como a resposta de uma solicitação HTTP

Explorar a linha do tempo do rastreamento

A seção Linha do tempo mostra uma hierarquia de segmentos e subsegmentos ao lado de uma barra horizontal que corresponde ao tempo que eles usaram para concluir suas tarefas. A primeira entrada na lista é o segmento, que representa todos os dados registrados pelo serviço para uma única solicitação. Os subsegmentos são indentados e listados após o segmento. As colunas contêm informações sobre cada segmento.

CloudWatch console

No console do CloudWatch, a Linha do tempo dos segmentos fornece as seguintes informações:

- A primeira coluna: lista os segmentos e subsegmentos no rastreamento selecionado.
- A coluna Status do segmento: lista o resultado do status de cada segmento e subsegmento.
- A coluna Código de resposta: lista um código de status de resposta HTTP para uma solicitação do navegador feita pelo segmento ou subsegmento, quando disponível.
- A coluna Duração: lista por quanto tempo o segmento ou subsegmento foi executado.
- A coluna Hospedado em: lista o namespace ou o ambiente em que o segmento ou subsegmento é executado, se aplicável. Para obter mais informações, consulte [Dimensões coletadas e combinações de dimensões](#).
- A última coluna: exibe barras horizontais que correspondem à duração da execução do segmento ou subsegmento, em relação aos outros segmentos ou subsegmentos na linha do tempo.

Para agrupar a lista de segmentos e subsegmentos por nó de serviço, ative Agrupar por nós.

X-Ray console

Na página de detalhes do rastreamento, escolha a guia Linha do tempo para ver a linha do tempo de cada segmento e subsegmento que compõe um rastreamento.

No console do X-Ray, a Linha do tempo fornece as seguintes informações:

- A coluna Nome: lista os nomes dos segmentos e subsegmentos no rastreamento.
- A coluna Res.: lista um código de status de resposta HTTP para uma solicitação do navegador feita pelo segmento ou subsegmento, quando disponível.
- A coluna Duração: lista por quanto tempo o segmento ou subsegmento foi executado.
- A coluna Status: lista o resultado do status do segmento ou subsegmento.
- A última coluna: exibe barras horizontais que correspondem à duração da execução do segmento ou subsegmento, em relação aos outros segmentos ou subsegmentos na linha do tempo.

Para ver os dados de rastreamento brutos que o console usa para gerar a linha do tempo, escolha a guia Dados brutos. Os dados brutos mostram informações sobre o rastreamento e os segmentos e subsegmentos que compõem o rastreamento no formato JSON. Essas informações podem incluir o seguinte:

- Carimbos de data/hora
- IDs exclusivos
- Recursos associados ao segmento ou subsegmento
- A fonte ou origem do segmento ou subsegmento
- Informações adicionais sobre a solicitação à sua aplicação, como a resposta de uma solicitação HTTP.

Quando você usa um SDK instrumentado da AWS, HTTP ou cliente SQL para fazer chamadas para recursos externos, o SDK do X-Ray registra os subsegmentos automaticamente. Você também pode usar o SDK do X-Ray para registrar subsegmentos personalizados para qualquer função ou bloco de código. Subsegmentos adicionais registrados enquanto um subsegmento personalizado estiver aberto tornam-se filhos do subsegmento personalizado.

Visualizar os detalhes do segmento

Na Linha do tempo do rastreamento, escolha o nome de um segmento para visualizar seus detalhes.

O painel de Detalhes do segmento mostra as guias Visão geral, Recursos, Anotações, Metadados, Exceções e SQL. O seguinte se aplica:

- A guia Overview (Visão geral) mostra informações sobre a solicitação e a resposta. As informações incluem o nome, a hora de início, a hora de término, a duração, o URL da solicitação, a operação da solicitação, o código de resposta da solicitação e quaisquer erros e falhas.
- A guia Recursos de um segmento mostra informações sobre o SDK do X-Ray e os recursos da AWS em execução na aplicação. Use os plug-ins Amazon EC2, AWS Elastic Beanstalk ou Amazon ECS para o SDK do X-Ray para registrar informações de recursos específicas do serviço. Para obter mais informações sobre plug-ins, consulte a seção Plug-ins de serviço em [Configurar o X-Ray SDK para Java](#).
- As guias restantes mostram Anotações, Metadados e Exceções registrados no segmento. As exceções são capturadas automaticamente quando geradas de uma solicitação instrumentada. As anotações e os metadados contêm informações adicionais que são registradas usando as operações fornecidos pelo SDK do X-Ray. Para adicionar anotações ou metadados a seus segmentos, use o SDK do X-Ray. Para obter mais informações, consulte o link específico do idioma listado em Instrumentar sua aplicação com SDKs do AWS X-Ray em [Instrumentando seu aplicativo para AWS X-Ray](#).

Visualizar os detalhes do subsegmento

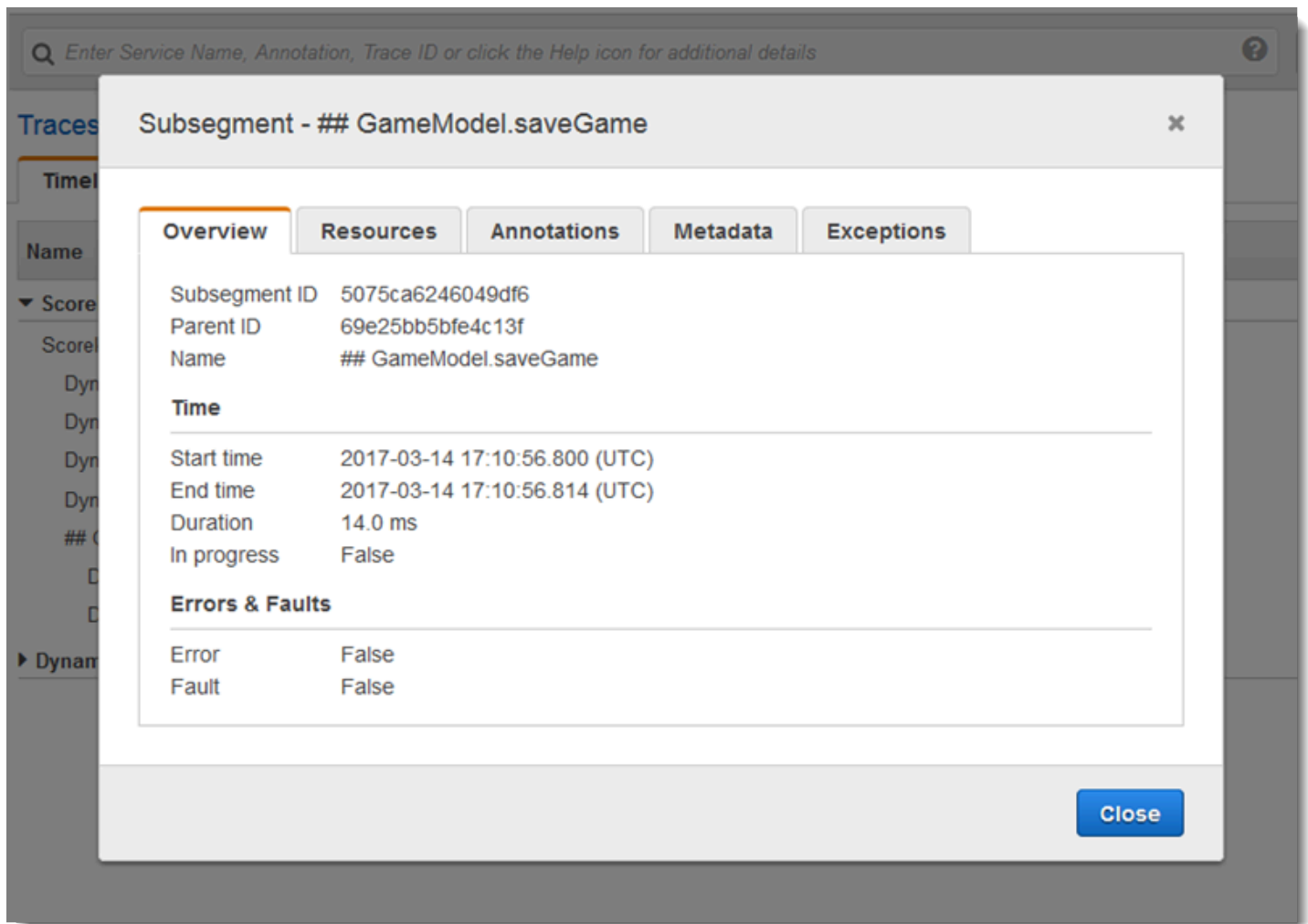
Na linha do tempo do rastreamento, escolha o nome de um subsegmento para visualizar os respectivos detalhes.

- A guia Visão geral contém informações sobre a solicitação e a resposta. Isso inclui o nome, a hora de início, a hora de término, a duração, o URL da solicitação, a operação da solicitação, o código de resposta da solicitação e quaisquer erros e falhas. Para subsegmentos gerados com clientes instrumentados, a guia Overview (Visão geral) contém informações sobre a solicitação e a resposta do ponto de vista do seu aplicativo.
- A guia Recursos de um subsegmento mostra detalhes sobre os recursos da AWS que foram usados para executar o subsegmento. Por exemplo, a guia de recursos pode incluir um ARN de função do AWS Lambda, informações sobre uma tabela do DynamoDB, qualquer operação chamada e o ID da solicitação.

- As guias restantes mostram Anotações, Metadados e Exceções registrados no subsegmento. As exceções são capturadas automaticamente quando geradas de uma solicitação instrumentada. As anotações e os metadados contêm informações adicionais que são registradas usando as operações fornecidas pelo SDK do X-Ray. Use o SDK do X-Ray para adicionar anotações ou metadados a seus segmentos. Para obter mais informações, consulte o link específico do idioma listado em Instrumentar sua aplicação com SDKs do AWS X-Ray em [Instrumentando seu aplicativo para AWS X-Ray](#).

Para subsegmentos personalizados, a guia Overview (Visão geral) mostra o nome do subsegmento, que é possível definir para especificar a área do código ou a função que ele registra. Para obter mais informações, consulte o link específico do idioma listado em Instrumentar sua aplicação com SDKs do AWS X-Ray em [Gerar subsegmentos personalizados com o X-Ray SDK para Java](#).

A imagem a seguir mostra a guia Visão geral de um subsegmento personalizado. A visão geral contém o ID do subsegmento, o ID principal, o nome, os horários de início e término, a duração, o status e os erros ou falhas.



A guia Metadados de um subsegmento personalizado contém informações no formato JSON sobre os recursos usados por esse subsegmento.

Usar expressões de filtro

Use expressões de filtro para visualizar rastreamentos ou um mapa de rastreamento para uma solicitação específica, um serviço, uma conexão entre dois serviços (uma borda) ou solicitações que satisfazem uma condição. O X-Ray fornece uma linguagem de expressão de filtro para filtrar solicitações, serviços e bordas com base em dados em cabeçalhos de solicitação, status de resposta e campos indexados nos segmentos originais.

Ao escolher um período de rastreamento para visualizar no console do X-Ray, você pode obter mais resultados do que o console é capaz de exibir. No canto superior direito, o console mostra o número de rastreamentos que verificou e se há mais rastreamentos disponíveis. É possível usar uma expressão de filtro para estreitar os resultados a apenas rastreamentos que você deseja localizar.

Tópicos

- [Detalhes de expressão de filtro](#)
- [Usar expressões de filtro com grupos](#)
- [Sintaxe de expressão de filtro](#)
- [Palavras-chave booleanas](#)
- [Palavras-chave de número](#)
- [Palavras-chave de string](#)
- [Palavras-chave complexas](#)
- [Função id](#)

Detalhes de expressão de filtro

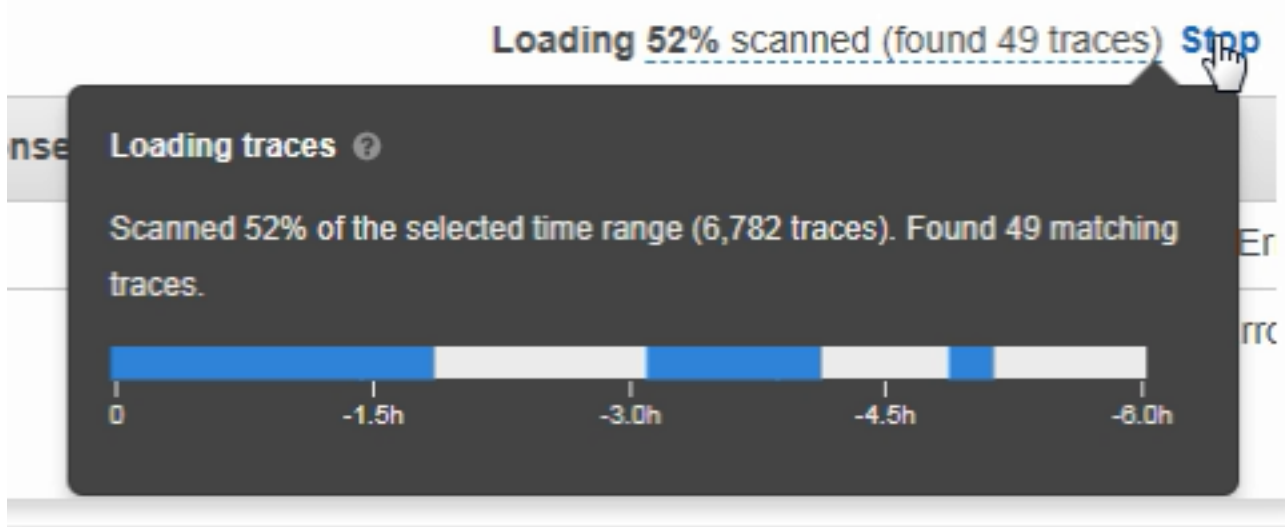
Quando você [escolhe um nó no mapa de rastreamento](#), o console cria uma expressão de filtro com base no nome do serviço do nó e nos tipos de erro presentes com base em sua seleção. Para encontrar rastreamentos que mostram problemas de desempenho ou relacionados a solicitações específicas, é possível ajustar a expressão fornecida pelo console ou criar a sua própria. Se você adicionar anotações com o X-Ray SDK, poderá também filtrar com base na presença de uma chave de anotação ou no valor de uma chave.

Note

Se você escolher um intervalo de tempo relativo no mapa de rastreamento e escolher um nó, o console converterá o intervalo de tempo em uma hora de início e de término absoluta. Para garantir que os rastreamentos do nó apareçam nos resultados da pesquisa e evitar tempos de verificação quando o nó não estava ativo, o intervalo de tempo inclui apenas as horas em que o nó enviou rastreamentos. Para pesquisar com relação à hora atual, é possível voltar a um intervalo de tempo relativo na página de rastreamentos e verificar novamente.

Se ainda houver mais resultados disponíveis além do que o console pode mostrar, o console mostrará o número de rastreamentos correspondidos e o número de rastreamentos verificados. A porcentagem mostrada é a porcentagem do intervalo de tempo selecionado que foi verificado. Para garantir que você veja todos os rastreamentos correspondentes representados nos resultados, restrinja mais sua expressão de filtro ou escolha um limite de tempo mais curto.

Para obter os resultados mais recentes primeiro, o console inicia a verificação no final do intervalo de tempo e trabalha retroativamente. Se houver um grande número de rastreamentos, mas poucos resultados, o console dividirá o intervalo de tempo em partes e os verificará em paralelo. A barra de progresso mostra as partes do intervalo de tempo que foram verificadas.



Usar expressões de filtro com grupos

Os grupos são uma coleção de rastreamentos definidos por uma expressão de filtro. Você pode usar grupos para gerar gráficos de serviço adicionais e fornecer métricas do Amazon CloudWatch.

Os grupos são identificados pelo nome ou pelo nome do recurso da Amazon (ARN) e contêm uma expressão de filtro. O serviço compara os rastreamentos de entrada com a expressão e os armazena adequadamente.

É possível criar e modificar grupos usando o menu suspenso à esquerda da barra de pesquisa da expressão de filtro.

Note

Se o serviço encontrar um erro na qualificação de um grupo, esse grupo não estará mais incluído no processamento de rastreamentos de entrada e uma métrica de erro será registrada.

Para obter mais informações sobre grupos, consulte [Configurar grupos](#).

Sintaxe de expressão de filtro

As expressões de filtro podem conter uma palavra-chave, um operador unário ou binário e um valor para comparação.

```
keyword operator value
```

Operadores diferentes estão disponíveis para tipos diferentes de palavras-chave. Por exemplo, `responsetime` é uma palavra-chave de número e pode ser comparada com os operadores relacionados a números.

Example Exemplo: solicitações em que o tempo de resposta foi superior a 5 segundos

```
responsetime > 5
```

É possível combinar várias expressões em uma expressão composta usando os operadores AND ou OR.

Example Exemplo: solicitações em que a duração total foi de 5 a 8 segundos

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

Palavras-chave e operadores simples encontram problemas apenas no nível de rastreamento. Se ocorrer um erro de downstream, mas é controlado pelo seu aplicativo e não retornado ao usuário, uma pesquisa por `error` não o encontrará.

Para encontrar rastreamentos com problemas de downstream, use as [palavras-chave complexas](#) `service()` e `edge()`. Essas palavras-chave permitem que você aplique uma expressão de filtro a todos os nós de downstream, um único nó de downstream ou uma borda entre dois nós. Para obter mais granularidade, filtre os serviços e bordas por tipo com [a função `id\(\)`](#).

Palavras-chave booleanas

Os valores de palavras-chave booleanas são verdadeiros ou falsos. Use essas palavras-chave para encontrar rastreamentos que resultaram em erros.

Palavras-chave booleanas

- `ok`: o código de status da resposta foi 2XX Êxito.

- `error`: o código de status da resposta foi 4XX Erro do cliente.
- `throttle`: o código de status da resposta foi 429 Solicitações em excesso.
- `fault`: o código de status da resposta foi 5XX Erro do servidor.
- `partial`: a solicitação tem segmentos incompletos.
- `inferred`: a solicitação inferiu segmentos.
- `first`: o elemento é o primeiro de uma lista enumerada.
- `last`: o elemento é o último de uma lista enumerada.
- `remote`: a entidade de causa raiz é remota.
- `root`: o serviço é o ponto de entrada ou o segmento raiz de um rastreamento.

Os operadores booleanos encontram segmentos onde a chave especificada é `true` ou `false`.

Operadores booleanos

- `none`: a expressão é verdadeira se a palavra-chave for verdadeira.
- `!`: a expressão é verdadeira se a palavra-chave for falsa.
- `=, !=`: compara o valor da palavra-chave com a string `true` ou `false`. Estes operadores agem da mesma forma que os outros operadores, mas são mais explícitos.

Example Exemplo: o status de resposta é 2XX OK

```
ok
```

Example Exemplo: o status de resposta não é 2XX OK

```
!ok
```

Example Exemplo: o status de resposta não é 2XX OK

```
ok = false
```

Example Exemplo: o último rastreamento de falha enumerado tem o nome de erro "deserialize"

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

Example Exemplo: solicitações com segmentos remotos em que a cobertura é maior que 0,7 e o nome do serviço é "traces"

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

Example Exemplo: solicitações com segmentos inferidos em que o tipo de serviço é "AWS:DynamoDB"

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

Example Exemplo: solicitações que têm um segmento com o nome "data-plane" como raiz

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

Palavras-chave de número

Use palavras-chave de número para pesquisar solicitações com um tempo de resposta, duração ou status de resposta específico.

Palavras-chave de número

- `responsetime`: o tempo que o servidor levou para enviar uma resposta.
- `duration`: duração total da solicitação, incluindo todas as chamadas subsequentes.
- `http.status`: código de status da resposta.
- `index`: posição de um elemento em uma lista enumerada.
- `coverage`: porcentagem decimal do tempo de resposta da entidade sobre o tempo de resposta do segmento raiz. Aplicável apenas para entidades de causa raiz de tempo de resposta.

Operadores de número

Palavras-chave de número usam operadores padrão de igualdade e comparação.

- `=, !=`: a palavra-chave é igual ou não a um valor numérico.
- `<, <=, >, >=`: a palavra-chave é menor ou maior do que um valor numérico.

Example Exemplo: o status de resposta não é 200 OK

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

Example Exemplo: solicitação em que a duração total foi de 5 a 8 segundos

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

Example Exemplo: solicitações que foram concluídas com êxito em menos de 3 segundos, incluindo todas as chamadas subsequentes

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

Example Exemplo: entidade de lista enumerada que tem um índice maior que 5

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

Example Exemplo: solicitações em que a última entidade tem cobertura superior a 0,8

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

Palavras-chave de string

Use palavras-chave de string para encontrar rastreamentos com texto específico em cabeçalhos de solicitação ou IDs de usuário específicos.

Palavras-chave de string

- `http.url`: URL da solicitação.
- `http.method`: método da solicitação.
- `http.useragent`: string do agente de usuário da solicitação.
- `http.clientip`: endereço IP do solicitante.
- `user`: valor do campo de usuário em qualquer segmento no rastreamento.
- `name`: o nome de um serviço ou exceção.
- `type`: tipo de serviço.
- `message`: mensagem de exceção.

- `availabilityzone`: valor do campo da zona de disponibilidade em qualquer segmento no rastreamento.
- `instance.id`: valor do campo de ID da instância em qualquer segmento no rastreamento.
- `resource.arn`: valor do campo de ARN do recurso em qualquer segmento no rastreamento.

Os operadores de string encontram valores que são iguais a ou contêm um texto específico. Os valores devem sempre ser especificados entre aspas.

Operadores de string

- `=,!=`: a palavra-chave é igual ou não a um valor numérico.
- `CONTAINS`: a palavra-chave contém uma string específica.
- `BEGINSWITH,ENDSWITH`: a palavra-chave começa ou termina com uma string específica.

Example Exemplo: filtro `http.url`

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

Para testar se um campo existe em um rastreamento, independentemente do seu valor, verifique se ele contém a string vazia.

Example Exemplo: filtro do usuário

Encontre todos os rastreamentos com IDs de usuário.

```
user CONTAINS ""
```

Example Exemplo: selecionar rastreamentos com uma causa raiz de falha que inclua um serviço chamado "Auth"

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

Example Exemplo: selecionar rastreamentos com uma causa raiz de tempo de resposta cujo último serviço tenha um tipo do DynamoDB

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

Example Exemplo: selecionar rastreamentos com uma causa raiz de falha cuja última exceção tenha a mensagem "Access Denied for account_id: 1234567890"

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

Palavras-chave complexas

Use palavras-chave complexas para encontrar solicitações com base no nome do serviço, no nome de borda ou no valor de anotação. Para serviços e bordas, você pode especificar uma expressão de filtro adicional que se aplica ao serviço ou borda. Para anotações, é possível filtrar pelo valor de uma anotação com uma chave específica usando operadores boolianos, de número ou de string.

Palavras-chave complexas

- `annotation[key]`: valor de uma anotação com o campo *key*. O valor de uma anotação pode ser um booliano, um número ou uma string; portanto, é possível usar qualquer um desses tipos de operador de comparação. É possível usar essa palavra-chave em combinação com as palavras-chave `service` ou `edge`. Uma chave de anotação que contenha pontos (pontos finais) deve ser colocada entre colchetes ([]).
- `edge(source, destination) {filter}`: conexão entre serviços *source* e *destination*. As chaves opcionais podem conter uma expressão de filtro que se aplica a segmentos nessa conexão.
- `group.name / group.arn`: o valor da expressão de filtro de um grupo, referido pelo nome do grupo ou ARN do grupo.
- `json`: objeto de causa raiz do JSON. Consulte [Obter dados do AWS X-Ray](#) para conhecer as etapas a fim de criar entidades JSON programaticamente.
- `service(name) {filter}`: serviço com *name*. Chaves opcionais podem conter uma expressão de filtro que se aplica a segmentos criados pelo serviço.

Use o serviço de palavras-chave para encontrar rastreamentos para solicitações que chegam a um determinado nó em seu mapa de rastreamento.

Os operadores complexos de palavras-chave encontram segmentos nos quais a chave especificada foi definida ou não.

Operadores de palavras-chave complexos

- `none`: a expressão é verdadeira se a palavra-chave for definida. Se a palavra-chave for do tipo booleano, ela será avaliada como o valor booleano.
- `!`: a expressão é verdadeira se a palavra-chave não for definida. Se a palavra-chave for do tipo booleano, ela será avaliada como o valor booleano.
- `=, !=`: compara o valor da palavra-chave.
- `edge(source, destination) {filter}`: conexão entre serviços *source* e *destination*. As chaves opcionais podem conter uma expressão de filtro que se aplica a segmentos nessa conexão.
- `annotation[key]`: valor de uma anotação com o campo *key*. O valor de uma anotação pode ser um booleano, um número ou uma string; portanto, é possível usar qualquer um desses tipos de operador de comparação. É possível usar essa palavra-chave em combinação com as palavras-chave `service` ou `edge`.
- `json`: objeto de causa raiz do JSON. Consulte [Obter dados do AWS X-Ray](#) para conhecer as etapas a fim de criar entidades JSON programaticamente.

Use o serviço de palavras-chave para encontrar rastreamentos para solicitações que chegam a um determinado nó em seu mapa de rastreamento.

Exemplo: filtro de serviço

Solicitações que incluem uma chamada para `api.example.com` com uma falha (erro da série 500).

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

É possível excluir o nome do serviço para aplicar uma expressão de filtro para todos os nós no seu mapeamento de serviço.

Exemplo: filtro de serviço

As solicitações que causam uma falha em qualquer lugar no mapa de rastreamento.

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

A palavra-chave de borda aplica uma expressão de filtro a uma conexão entre dois nós.

Example Exemplo: filtro de borda

Solicitação em que o serviço `api.example.com` fez uma chamada para o `backend.example.com` que gerou um erro.

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

Use também o operador `!` com palavras-chave de serviço e borda para excluir um serviço ou borda dos resultados de outra expressão de filtro.

Example Exemplo: filtro de serviço e solicitação

Solicitação em que o URL começa com `http://api.example.com/` e contém `/v2/`, mas não abrange um serviço chamado `api.example.com`.

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

Example Exemplo: filtro de serviço e tempo de resposta

Encontre rastreamento em que `http url` está definido e o tempo de resposta é maior que 2 segundos.

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

Para anotações, você pode chamar todos os rastreamentos em que `annotation[key]` está definido ou usar os operadores de comparação que correspondem ao tipo de valor.

Example Exemplo: anotação com valor de string

Solicitações com uma anotação chamada `gameid` com o valor de string `"817DL6V0"`.

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

Example Exemplo: a anotação está definida

Solicitações com uma anotação definida como `age`.

```
annotation[age]
```

Example Exemplo: a anotação não está definida

Solicitações sem uma anotação definida como age.

```
!annotation[age]
```

Example Exemplo: anotação com valor numérico

Solicitações cuja idade de anotação tem um valor numérico maior do que 29.

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

Example Exemplo: anotação em combinação com serviço ou borda

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

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

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

Example Exemplo: grupo com usuário

Solicitações em que os rastreamentos atendem ao filtro de grupo high_response_time (por exemplo, responseTime > 3) e o usuário se chama Alice.

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

Example Exemplo: JSON com entidade de causa raiz

Solicitações com entidades de causa raiz correspondentes.

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

Função id

Ao fornecer um nome de serviço para as palavras-chave `service` ou `edge`, você obtém resultados para todos os nós que têm esse nome. Para filtragem mais precisa, use a função `id` para especificar um tipo de serviço, além de um nome para distinguir os nós com o mesmo nome.

Use a função `account.id` para especificar uma conta específica para o serviço ao visualizar rastreamentos de várias contas em uma conta de monitoramento.

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

Use a função `id` no lugar de um nome de serviço nos filtros de serviço e de borda.

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

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

Por exemplo, as funções do AWS Lambda resultam em dois nós no mapa de rastreamento; uma para a invocação da função e outro para o serviço Lambda. Os dois nós têm o mesmo nome, mas tipos diferentes. Um filtro de serviço padrão encontrará rastreamentos para ambos.

Example Exemplo: filtro de serviço

As solicitações que incluem um erro em qualquer serviço chamado `random-name`.

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

Use a função `id` para restringir a pesquisa para erros na função em si, excluindo os erros do serviço.

Example Exemplo: filtro de serviço com a função `id`

Solicitações que incluem um erro em um serviço chamado `random-name` com tipo `AWS::Lambda::Function`.

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

Para procurar nós por tipo, também é possível excluir o nome inteiramente.

Example Exemplo: filtro de serviço com função id e tipo de serviço

Solicitações que incluem um erro em um serviço com o tipo `AWS::Lambda::Function`.

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

Para pesquisar nós referentes a determinada Conta da AWS, especifique um ID de conta.

Example Exemplo: filtro de serviço com a função id e o ID da conta

Solicitações que incluem um serviço dentro de um ID de conta `AWS::Lambda::Function` específico.

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

Rastreamento entre contas

O AWS X-Ray permite a observabilidade entre contas, possibilitando que você monitore e solucione problemas de aplicações que abrangem várias contas em uma Região da AWS. Você pode pesquisar, visualizar e analisar facilmente métricas, logs e rastreamentos em qualquer conta vinculada, como se estivesse operando em uma única conta. Isso fornece uma visão completa das solicitações que se estendem por várias contas. Você pode visualizar rastreamentos entre contas no mapa de rastreamento do X-Ray e nas páginas de rastreamentos no [console do CloudWatch](#).

Os dados de observabilidade compartilhados podem incluir qualquer um dos seguintes tipos de telemetria:

- Métricas no Amazon CloudWatch
- Grupos de logs no Amazon CloudWatch Logs
- Rastreamentos no AWS X-Ray
- Aplicações no Amazon CloudWatch Application Insights

Configurar a observabilidade entre contas

Para ativar a observabilidade entre contas, configure uma ou mais contas de monitoramento da AWS e vincule-as a diversas contas de origem. Uma conta de monitoramento é uma Conta da AWS central que pode visualizar e interagir com dados de observabilidade gerados das contas de origem. Uma conta de origem é uma Conta da AWS individual que gera dados de observabilidade para os recursos contidos nela.

As contas de origem compartilham os dados de observabilidade com as contas de monitoramento. Os rastreamentos são copiados de cada conta de origem para até cinco contas de monitoramento. As cópias dos rastreamentos das contas de origem para a primeira conta de monitoramento são gratuitas. As cópias dos rastreamentos enviados a contas de monitoramento adicionais são cobradas em cada conta de origem, com base no preço padrão. Para obter mais informações, consulte [Preços do AWS X-Ray](#) e [Definição de preço do Amazon CloudWatch](#).

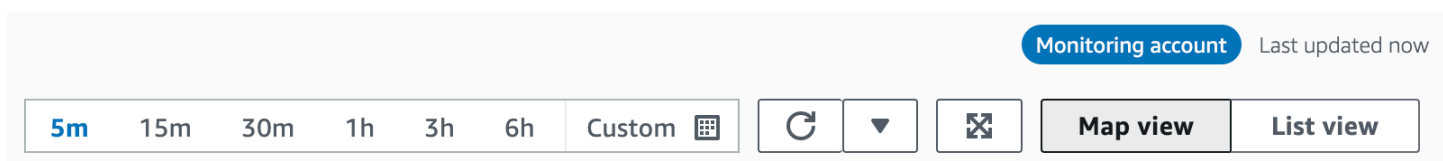
Para criar vínculos entre as contas de monitoramento e as contas de origem, use o console do CloudWatch ou os novos comandos do Observability Access Manager na AWS CLI e na API. Para obter mais informações, consulte [Observabilidade entre contas do CloudWatch](#).

Note

Os rastreamentos do X-Ray são cobrados na Conta da AWS em que são recebidos. Se uma solicitação [amostrada](#) abranger serviços em mais de uma Conta da AWS, cada conta registrará um rastreamento separado e todos os demais compartilharão o mesmo ID de rastreamento. Para saber mais sobre os preços de observabilidade entre contas, consulte [Preços do AWS X-Ray](#) e [Definição de preço do Amazon CloudWatch](#).

Visualizar rastreamentos entre contas

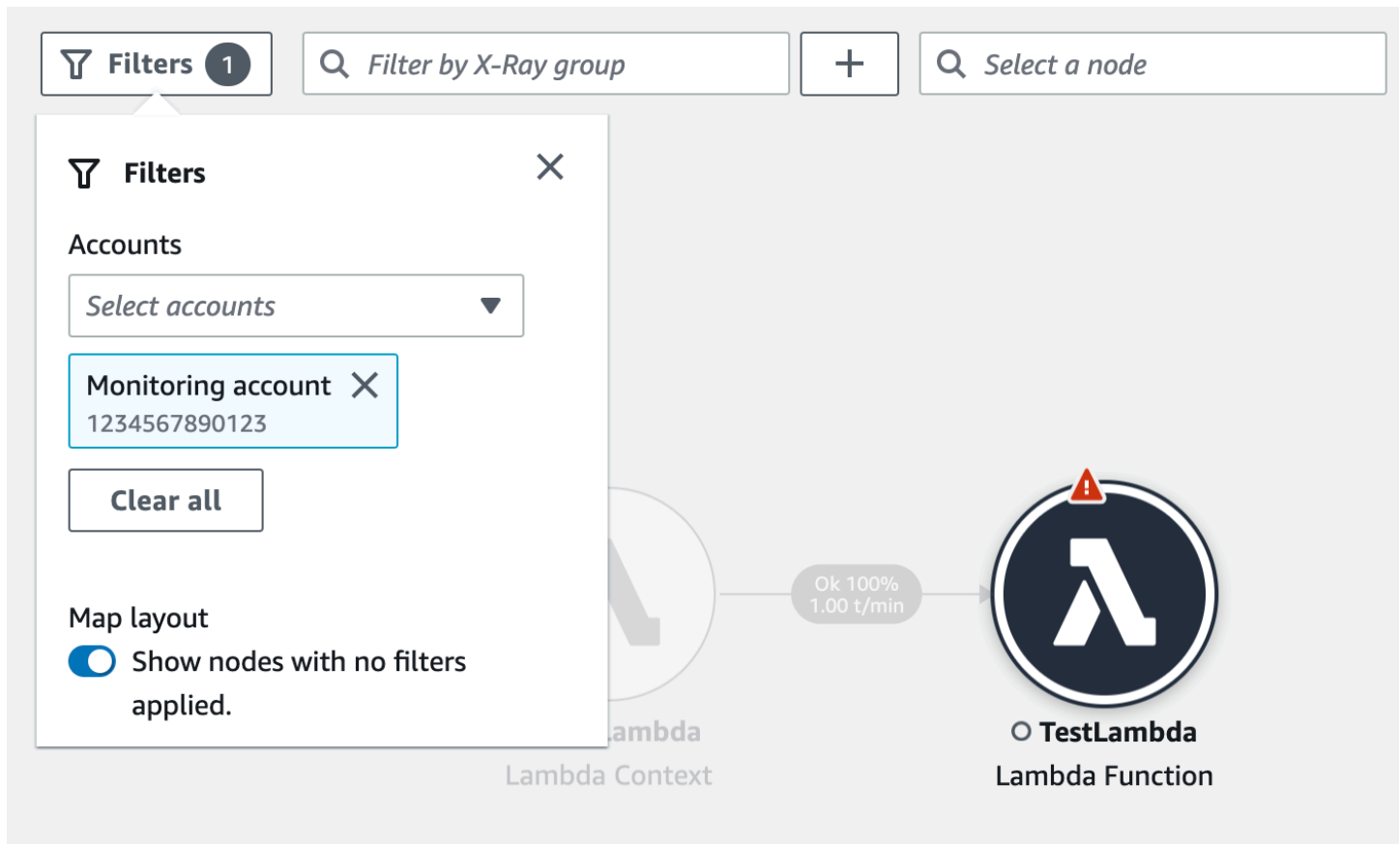
Os rastreamentos entre contas são exibidos na conta de monitoramento. Cada conta de origem exibe somente rastreamentos locais para essa conta específica. As seções a seguir pressupõem que você fez login na conta de monitoramento e abriu o console do Amazon CloudWatch. Tanto no mapa de rastreamento quanto nas páginas de rastreamentos, um selo da conta de monitoramento é exibido no canto superior direito.



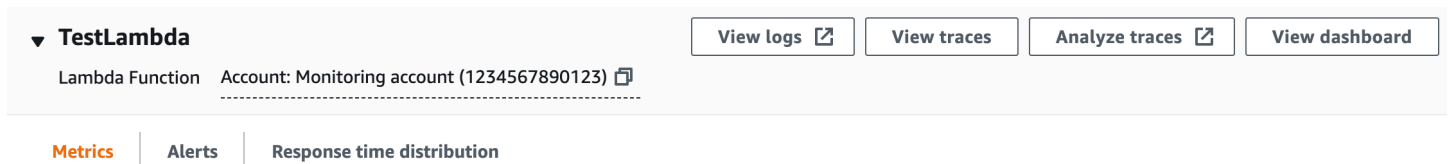
Mapa de rastreamento

No console do CloudWatch, escolha Mapa de rastreamento em Rastreamentos do X-Ray no painel de navegação esquerdo. Por padrão, o mapa de rastreamento exibe nós para todas as contas de origem que enviam rastreamentos à conta de monitoramento e nós para a própria conta de monitoramento. No mapa de rastreamento, escolha Filtros no canto superior esquerdo para filtrar

lo usando o menu suspenso Contas. Depois que um filtro de conta é aplicado, os nós de serviço de contas que não correspondem ao filtro atual ficam desabilitados.



Quando você escolhe um nó de serviço, o painel de detalhes do nó inclui o ID da conta e o rótulo do serviço.



No canto superior direito do mapa de rastreamento, escolha Visualização em lista para ver uma lista de nós de serviço. A lista de nós de serviço inclui serviços da conta de monitoramento e de todas as contas de origem configuradas. Filtre a lista de nós por Rótulo da conta ou ID da conta escolhendo-os no filtro Nós.

Nodes (2)

Account id = | X

Use: "Account id = "

Values

Account id = 461265027466

Alarms ▾	Latency (avg) ▾	Faults (5xx) ▾
 1	13ms	0.00/min

Rastreamentos

Visualize os detalhes do rastreamento que abrangem várias contas abrindo o console do CloudWatch na conta de monitoramento e escolhendo Rastreamentos em Rastreamentos do X-Ray no painel de navegação esquerdo. Você também pode abrir essa página escolhendo um nó no Mapa de rastreamento do X-Ray e selecionando Visualizar rastreamentos no painel de detalhes do nó.

A página Rastreamentos permite consultas por ID de conta. Para começar, [insira uma consulta](#) que inclua um ou mais IDs de conta. O exemplo a seguir consulta rastreamentos que passaram pelo ID da conta X ou Y:

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

Traces [Info](#)

5m 15m 30m 1h 3h 6h Custom 

Find traces by typing a query, build a query using the Query refiners section, or [choose a sample query](#). You can also [find a trace by ID](#).

Filter by X-Ray group

service(id(account.id: "1234567890123"))

Run query  5 traces retrieved

Refine sua consulta por Conta. Selecione uma ou mais contas na lista e escolha Adicionar à consulta.

▼ Query refiners

Refine query by

Account ▼

1 selected

Add to query

Select rows to filter traces

 Find Account name and ID

< 1 >



Account name and ID ▼



Monitoring account (1234567890123)

Detalhes de rastreamento

Veja os detalhes de um rastreamento escolhendo-o na lista Rastreamentos na parte inferior da página Rastreamentos. Os Detalhes de rastreamento são exibidos, incluindo um mapa de detalhes do rastreamento com nós de serviço de todas as contas pelas quais o rastreamento passou. Escolha um nó de serviço específico para ver a conta correspondente.

A seção Linha do tempo dos segmentos exibe os detalhes da conta para cada segmento na linha do tempo.

▼ TestLambda AWS::Lambda::Function Monitoring account (1234567890123)

TestLambda	✓ OK	-	28ms	
Invocation	✓ OK	-	1ms	
Overhead	✓ OK	-	8ms	

Rastrear aplicações orientadas a eventos

O AWS X-Ray é compatível com o rastreamento de aplicações orientados a eventos usando o Amazon SQS e o AWS Lambda. Use o console do CloudWatch para ver uma visualização conectada de cada solicitação à medida que ela é enfileirada com o Amazon SQS por uma ou mais funções do Lambda. Os rastreamentos dos produtores de mensagens precedentes são automaticamente vinculados aos rastreamentos dos nós consumidores subsequentes do Lambda, criando uma visão completa da aplicação.

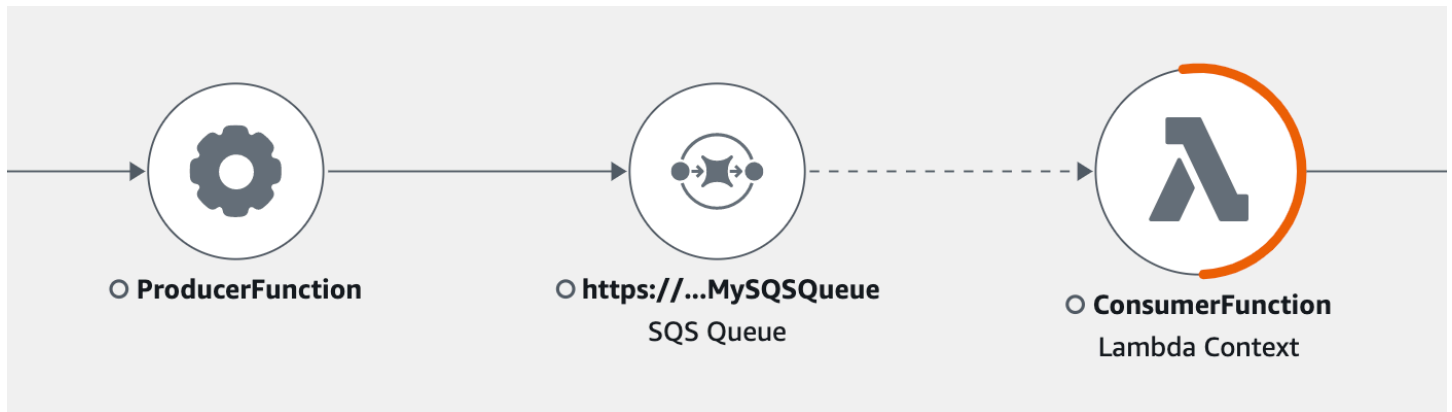
Note

Cada segmento de rastreamento pode ser vinculado a até vinte traços, enquanto um rastreamento pode incluir no máximo cem links. Em determinadas situações, ao vincular rastreamentos adicionais, o [tamanho máximo do documento de rastreamentos](#) pode ser

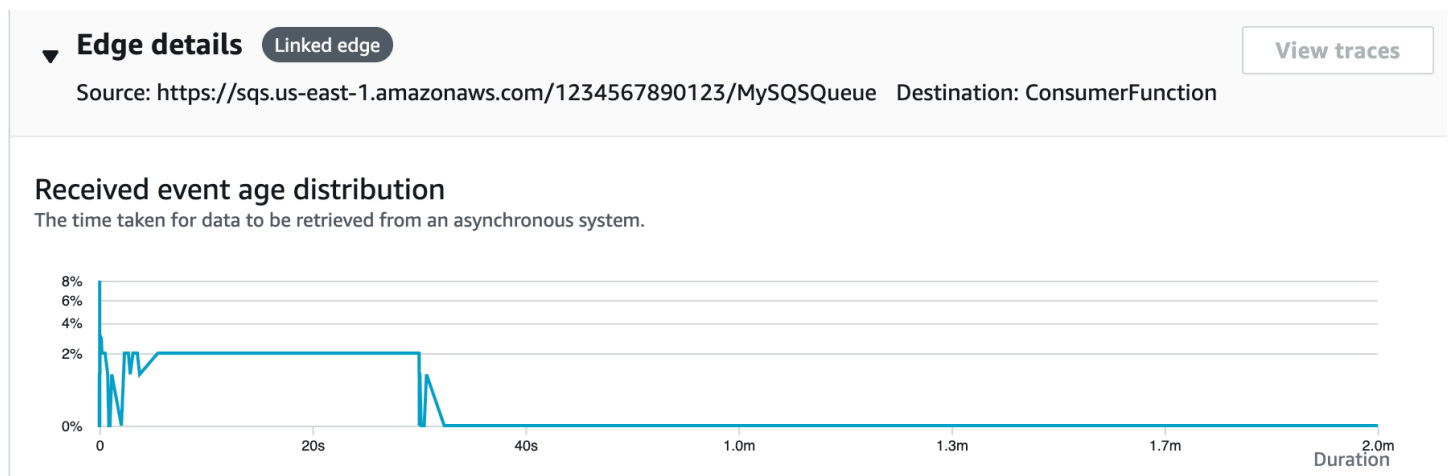
excedido e resultar em um rastreamento possivelmente incompleto. Isso pode acontecer, por exemplo, quando uma função do Lambda com rastreamento habilitado envia muitas mensagens SQS para uma fila em uma única invocação. Se você se deparar com esse problema, há um método de mitigação que usa os X-Ray SDKs. Consulte o X-Ray SDK para [Java](#), [Node.js](#), [Python](#), [Go](#) ou [.NET](#) para obter mais informações.

Visualizar rastreamentos vinculados no mapa de rastreamento

Use a página Mapa de rastreamento no [console do CloudWatch](#) para visualizar um mapa de rastreamento com rastreamentos de produtores de mensagens vinculados a rastreamentos de consumidores do Lambda. Esses links são exibidos com uma borda tracejada que conecta o nó do Amazon SQS e os nós consumidores subsequentes do Lambda.



Selecione uma borda tracejada para exibir um histograma da idade do evento recebido, que mapeia a distribuição da idade do evento quando ele é recebido pelos consumidores. A idade é calculada sempre que um evento é recebido.



Visualizar detalhes de rastreamentos vinculados

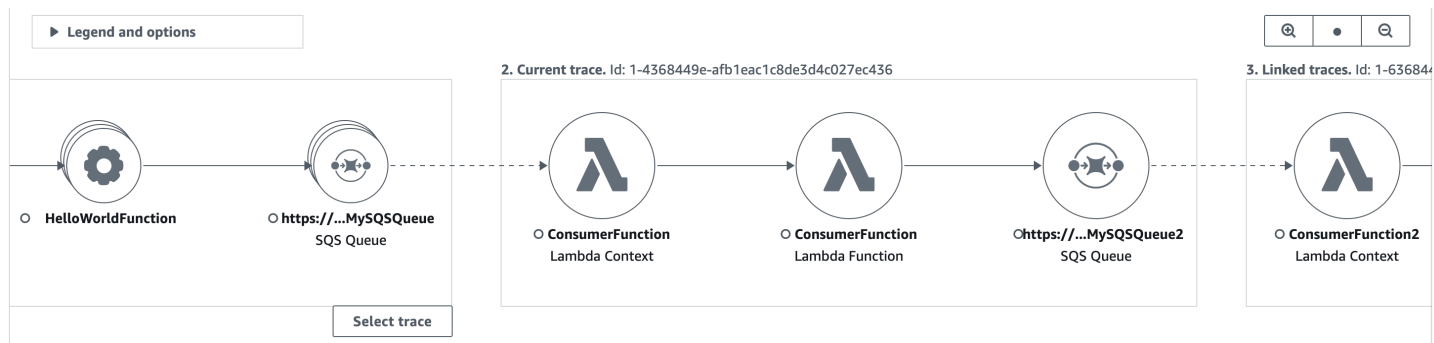
Veja os detalhes dos rastreamentos enviados por um produtor de mensagens, uma fila do Amazon SQS ou um consumidor do Lambda:

1. Use o mapa de rastreamento para selecionar um produtor de mensagens, o Amazon SQS ou nó consumidor do Lambda.
2. Escolha Visualizar rastreamentos no painel de detalhes do nó para exibir uma lista de rastreamentos. Você também pode navegar diretamente para a página Rastreamentos no console do CloudWatch.
3. Escolha um rastreamento específico na lista para abrir a página de detalhes do rastreamento. A página de detalhes do rastreamento exibe uma mensagem quando o rastreamento selecionado faz parte de um conjunto vinculado de rastreamentos.

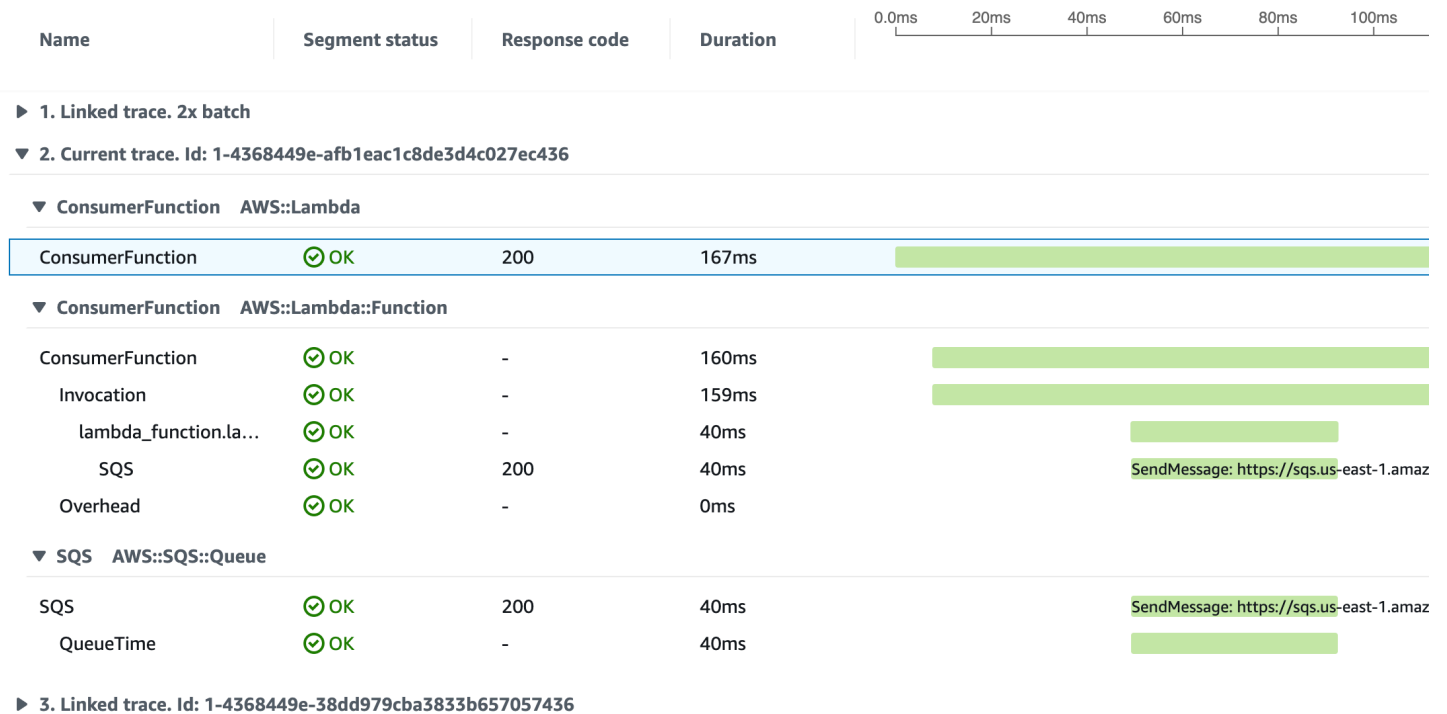
CloudWatch > Traces > Trace 1-4368449e-afb1eac1c8de3d4c027ec436

Trace 1-6368449e-afb1eac1c8de3d4c027ec436 Info This trace is part of a linked set of traces

O mapa de detalhes do rastreamento exibe o rastreamento atual, bem como os rastreamentos vinculados precedentes e subsequentes, cada um deles contido em uma caixa que indica os limites de cada rastreamento. Se o rastreamento selecionado no momento estiver vinculado a vários rastreamentos precedentes e subsequentes, os nós dentro dos rastreamentos vinculados precedentes e subsequentes serão empilhados e um botão Selecionar rastreamento será exibido.



Abaixo do mapa de detalhes do rastreamento, é exibida uma linha do tempo dos segmentos de rastreamento, incluindo rastreamentos vinculados precedentes e subsequentes. Se houver vários rastreamentos vinculados precedentes e subsequentes, os detalhes do segmento não poderão ser exibidos. Para visualizar os detalhes do segmento de um único rastreamento em um conjunto de rastreamentos vinculados, [selecione um único rastreamento](#) conforme descrito abaixo.

Segments Timeline [Info](#)

Selecionar um único rastreamento dentro de um conjunto de rastreamentos vinculados

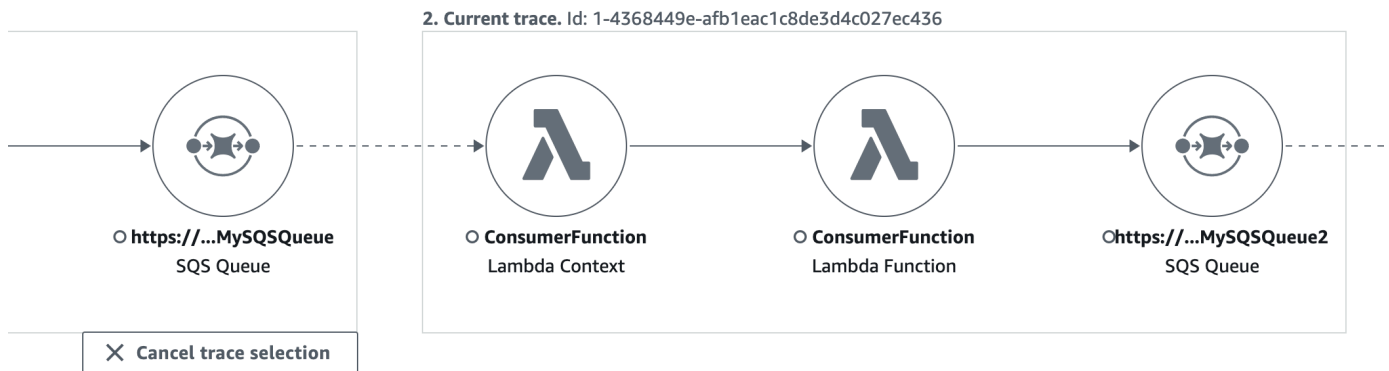
Filtre um conjunto vinculado de rastreamentos em um único rastreamento para ver os detalhes do segmento na linha do tempo.

- Escolha Selecionar rastreamento abaixo dos rastreamentos vinculados no mapa de detalhes do rastreamento. Uma lista de rastreamentos é exibida.

Traces (2)				
Start typing to filter trace list				
ID	Trace status	Timestamp	Response code	
☒ ...3fd6e9600d58fea82597e9af	✓ OK	11.7min (2022-11-06 15:34:54)	200	
☐ ...223d41cc17bae4a5394423a0	✓ OK	11.7min (2022-11-06 15:34:54)	200	

- Selecione o botão de opção ao lado de um rastreamento para visualizá-lo no mapa de detalhes do rastreamento.

3. Escolha Cancelar seleção de rastreamentos para visualizar todo o conjunto de rastreamentos vinculados.



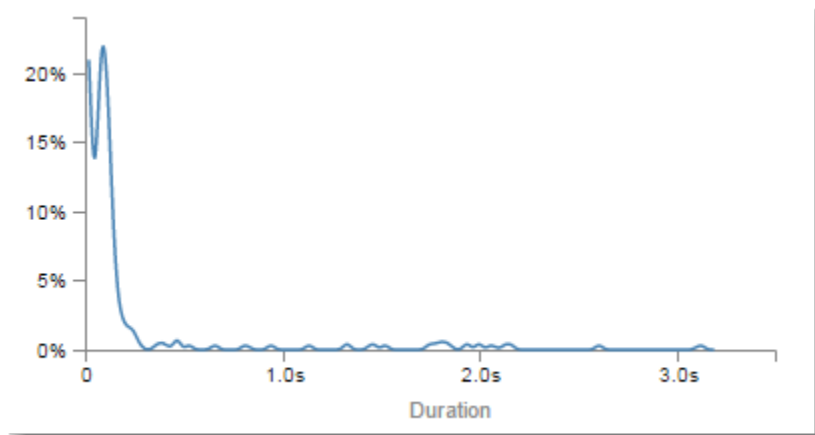
Usar histogramas de latência

Quando você seleciona um nó ou uma borda em um [mapa de rastreamento](#) do AWS X-Ray, o console do X-Ray mostra um histograma de distribuição da latência.

Latência

A latência é o tempo entre o início de uma solicitação e sua conclusão. Um histograma mostra uma distribuição de latências. Ela mostra a duração no eixo x e a porcentagem de solicitações correspondentes a cada período no eixo y.

Este histograma mostra um serviço que conclui a maioria das solicitações em menos de 300 ms. Uma pequena porcentagem de solicitações demora até 2 segundos, e algumas exceções levam mais tempo.



Interpretar detalhes do serviço

Os histogramas de serviço e margem apresentam uma representação visual da latência a partir do ponto de vista de um serviço ou solicitante.

- Clique no círculo para escolher um nó de serviço. O X-Ray mostra um histograma de solicitações atendidas pelo serviço. As latências são as registradas pelo serviço e não incluem nenhuma latência de rede entre o serviço e o solicitante.
- Escolha uma borda clicando na linha ou na ponta da seta da borda entre dois serviços. O X-Ray mostra um histograma para pedidos do solicitante que foram atendidos pelo serviço subsequente. As latências são as registradas pelo solicitante e incluem a latência na conexão de rede entre os dois serviços.

Para interpretar o histograma de painel Service details, é possível examinar os valores que mais diferem da maioria dos valores no histograma. Essas exceções podem ser vistas como picos ou picos no histograma, e é possível visualizar os rastreamentos de uma área específica para investigar o que está acontecendo.

Para visualizar rastreamentos filtrados por latência, selecione um intervalo no histograma. Clique onde você deseja iniciar a seleção e arraste da esquerda para a direita a fim de destacar um intervalo de latências a serem incluídas no filtro de rastreamento.

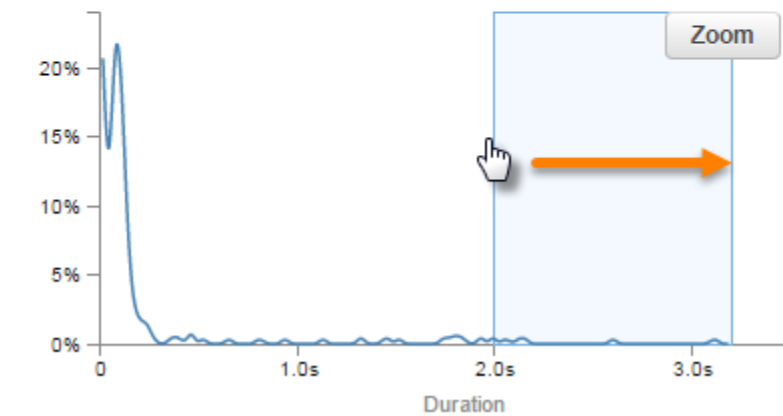
Service details ?

Name: Scorekeep

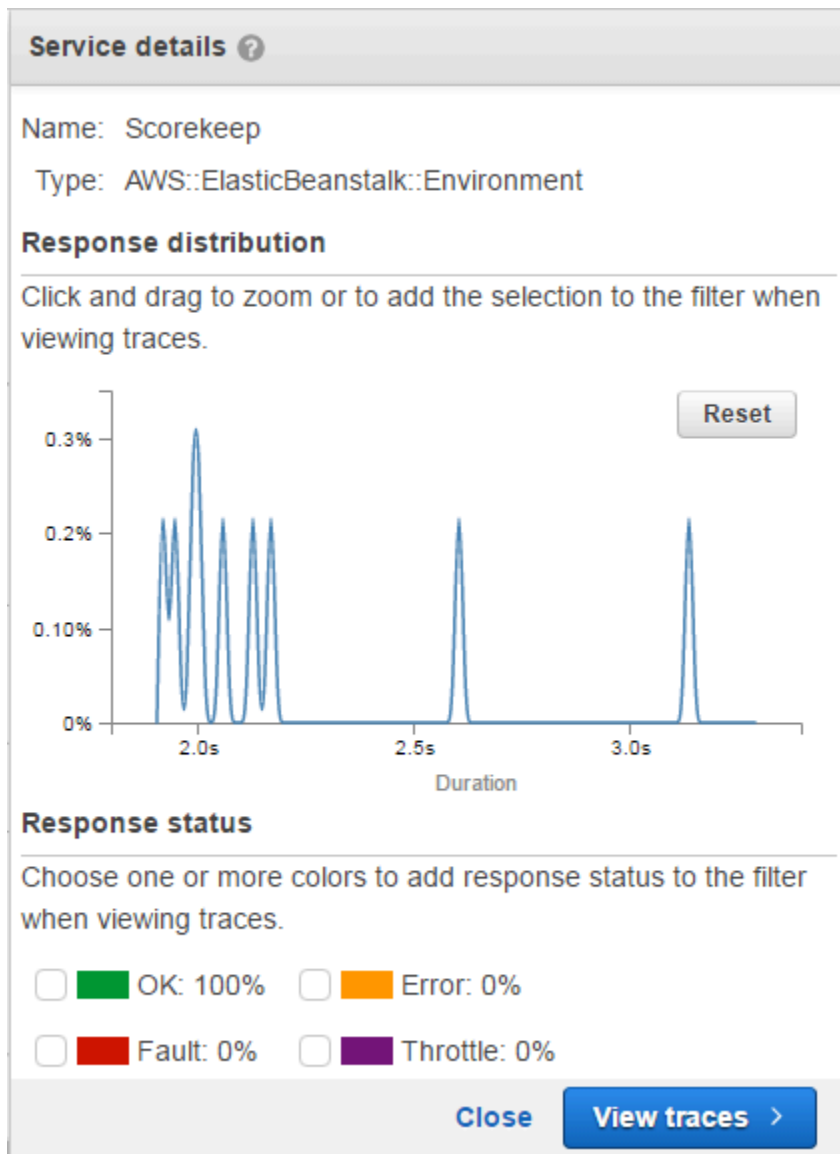
Type: AWS::ElasticBeanstalk::Environment

Response distribution

Click and drag to zoom or to add the selection to the filter when viewing traces.



Depois de selecionar um intervalo, você poderá escolher Zoom para visualizar apenas essa parte do histograma e refinar a seleção.



Assim que você tiver o foco definido na área pela qual tem interesse, escolha View traces.

Usar os insights do X-Ray

O AWS X-Ray analisa continuamente os dados de rastreamento em sua conta para identificar problemas emergentes nas aplicações. Quando as taxas de falha excedem o intervalo esperado, ele cria um insight que registra o problema e rastreia o respectivo impacto até que ele seja resolvido.

Com os insights, você pode:

- Identificar onde os problemas estão ocorrendo na aplicação, a causa raiz do problema e o impacto correspondente. A análise de impacto fornecida pelos insights permite que você determine a gravidade e a prioridade de um problema.

- Receber notificações à medida que o problema muda ao longo do tempo. As notificações de insights podem ser integradas à sua solução de monitoramento e alerta usando o Amazon EventBridge. Essa integração permite que você envie e-mails ou alertas automatizados com base na gravidade do problema.

O console do X-Ray identifica nós com incidentes contínuos no mapa de rastreamento. Para ver um resumo de insights, escolha o nó afetado. Você também pode visualizar e filtrar insights escolhendo Insights no painel de navegação à esquerda.



O X-Ray cria um insight quando detecta uma anomalia em um ou mais nós do mapa de serviço. O serviço usa modelagem estatística para prever as taxas de falha esperadas dos serviços na aplicação. No exemplo anterior, a anomalia é um aumento nas falhas do AWS Elastic Beanstalk. O servidor do Elastic Beanstalk teve vários tempos limite de chamadas de API, causando uma anomalia nos nós subsequentes.

Habilitar insights no console X-Ray

É necessário habilitar os insights para cada grupo com o qual você deseja usar os recursos de insight. Você pode habilitá-los na página Grupos.

1. Abra o [console do X-Ray](#).

2. Selecione um grupo ou crie um escolhendo Criar grupo e escolha Habilitar Insights. Para obter mais informações sobre como configurar grupos no console do X-Ray, consulte [Configurar grupos](#).
3. No painel de navegação à esquerda, escolha Insights e selecione um insight para visualizar.

From	2021-01-18 20:00	To	2021-01-20 11:00	Group	Default	State	All	Apply filter		
Description	Duration	Root cause service	Anomalous services	Group	Start time					
Overall, 30% of the client requests failed due to faults and 19% of the requests to api (AWS::ElasticBeanstalk::Environment) failed due to faults. Closed Fault	2 minutes 58 seconds	api (AWS::ElasticBeanstalk::Envir...	www (AWS::ElasticBeanstalk::Envir... api (AWS::ElasticBeanstalk::Envir...	Default	Jan 19th 2021, 19:02					

Note

O X-Ray usa as operações de API GetInsightSummaries, GetInsight, GetInsightEvents e GetInsightImpactGraph para recuperar dados de insights.

Para obter mais informações, consulte [Como o AWS X-Ray funciona com o IAM](#).

Habilitar notificações de insights

Com as notificações de insights, uma notificação é criada para cada evento de insight, como quando um insight é criado, muda significativamente ou é fechado. Os clientes podem receber essas notificações por meio de eventos do Amazon EventBridge e usar regras condicionais para realizar ações como notificação do SNS, invocação do Lambda e publicação de mensagens em uma fila do SQS ou qualquer um dos destinos compatíveis com o EventBridge. As notificações de insight são emitidas com base no melhor esforço, mas não são garantidas. Para obter mais informações sobre destinos, consulte [Amazon EventBridge Targets](#).

Você pode habilitar as notificações de insight para qualquer grupo habilitado para insights na página Grupos.

Como habilitar notificações para um grupo do X-Ray

1. Abra o [console do X-Ray](#).
2. Selecione um grupo ou crie um escolhendo Criar grupo. Certifique-se de que Habilitar Insights esteja selecionado e escolha Habilitar notificações. Para obter mais informações sobre como configurar grupos no console do X-Ray, consulte [Configurar grupos](#).

Como configurar regras condicionais do Amazon EventBridge

1. Abra o [console do Amazon EventBridge](#).
2. Navegue até Regras na barra de navegação esquerda e escolha Criar regra.
3. Forneça um nome e uma descrição para a regra.
4. Escolha Padrão de evento e selecione Padrão personalizado. Forneça um padrão contendo "source": ["aws.xray"] e "detail-type": ["AWS X-Ray Insight Update"]. Veja a seguir alguns exemplos de padrões possíveis.
 - Padrão de evento para corresponder a todos os eventos de entrada de insights do X-Ray:

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

- Padrão de evento para corresponder a um **state** e **category** específicos:

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

5. Selecione e configure os destinos que você gostaria de invocar quando um evento corresponder a essa regra.
6. (Opcional) Forneça tags para identificar e selecionar essa regra com maior facilidade.
7. Escolha Criar.

Note

As notificações de insights do X-Ray enviam eventos para o Amazon EventBridge, que no momento não permite chaves gerenciadas pelo cliente. Para obter mais informações, consulte [Proteção de dados no AWS X-Ray](#).

Visão geral do insight

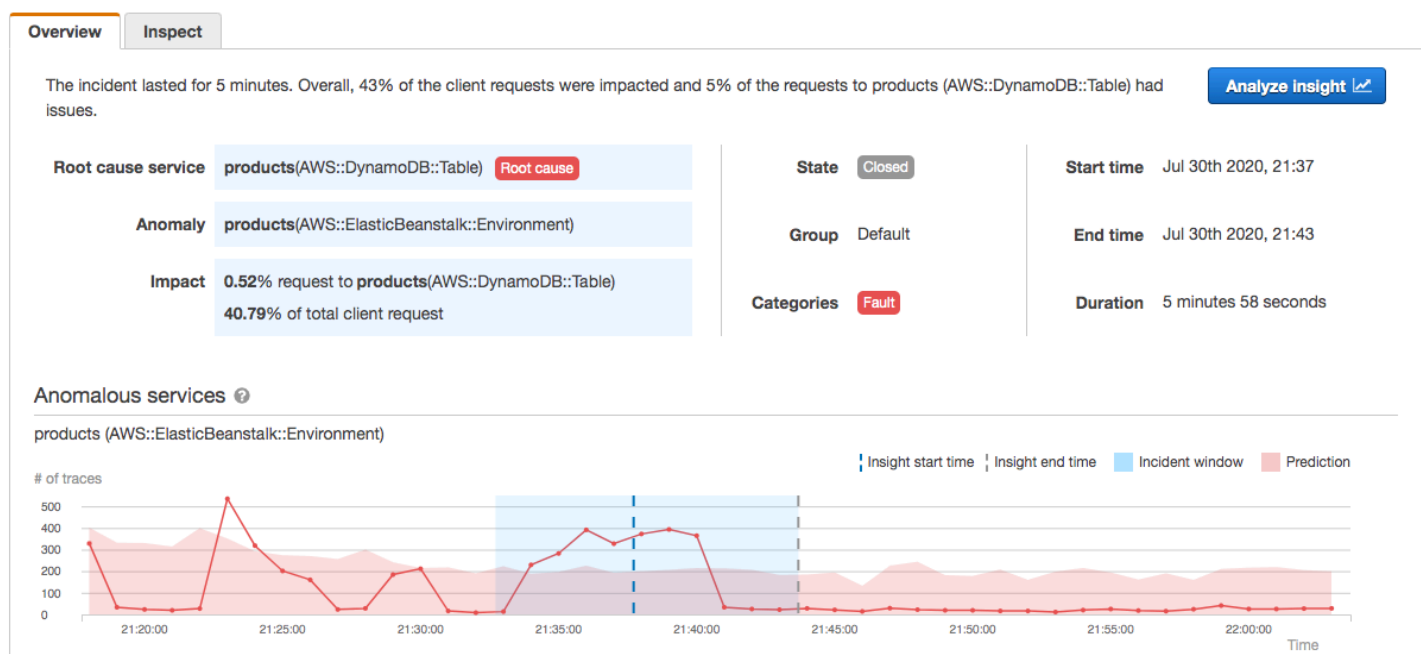
A página de visão geral de um insight tenta responder a três perguntas principais:

- Qual é o problema subjacente?
- Qual é a causa raiz?
- Qual é o impacto?

A seção Serviços anômalos mostra um cronograma para cada serviço que ilustra a mudança nas taxas de falha durante o incidente. A linha do tempo mostra o número de rastreamentos com falhas sobrepostas em uma faixa sólida que indica o número esperado de falhas com base na quantidade de tráfego registrada. A duração do insight é visualizada por meio da Janela de incidentes. A janela de incidentes começa quando o X-Ray observa a métrica se tornando anômala e persiste enquanto o insight está ativo.

O exemplo a seguir mostra um aumento nas falhas que causaram um incidente:

products (AWS::DynamoDB::Table) of Default group



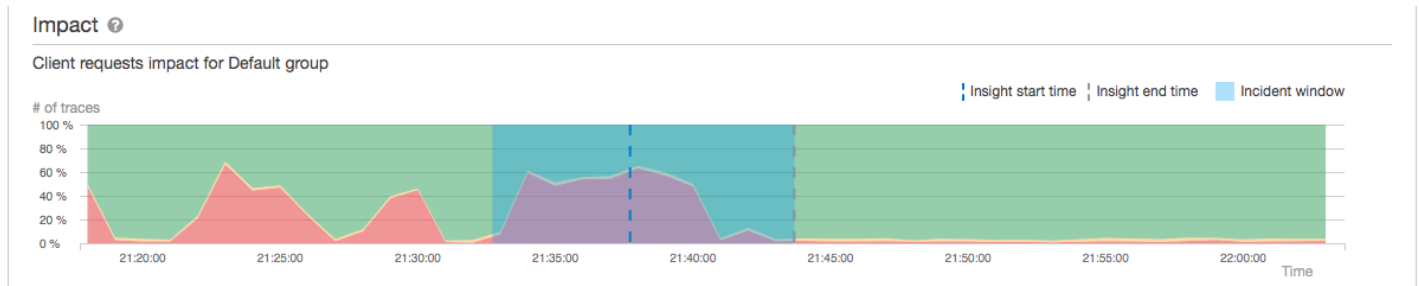
A seção Causa raiz mostra um mapa de rastreamento focado no serviço de causa raiz e no caminho afetado. É possível ocultar os nós não afetados selecionando o ícone de olho na parte superior direita do mapa da causa raiz. O serviço de causa raiz é o nó posterior mais distante em que o X-Ray identificou uma anomalia. Ele pode representar um serviço que você instrumentou ou um serviço externo que seu serviço chamou com um cliente instrumentado. Por exemplo, se você chamar o

Amazon DynamoDB com um cliente de SDK da AWS instrumentado, um aumento nas falhas do DynamoDB resultará em um insight com o DynamoDB como causa raiz.

Para investigar melhor a causa raiz, selecione Visualizar detalhes da causa raiz no gráfico da causa raiz. Você pode usar a página Análises para investigar a causa raiz e as mensagens relacionadas. Para obter mais informações, consulte [Interagir com o console do Analytics](#).



As falhas que continuam na parte precedente no mapa podem afetar vários nós e causar várias anomalias. Se uma falha for passada de volta para o usuário que fez a solicitação, o resultado será uma falha do cliente. Essa é uma falha no nó raiz do mapa de rastreamento. O gráfico Impacto fornece uma linha do tempo da experiência do cliente para todo o grupo. Essa experiência é calculada com base em porcentagens dos seguintes estados: Falha, Erro, Controle e OK.



Este exemplo mostra um aumento nos rastreamentos com uma falha no nó raiz durante o período de um incidente. Incidentes em serviços subsequentes nem sempre correspondem a um aumento nos erros de cliente.

Escolher Analisar insight abre o console do X-Ray Analytics em uma janela na qual você pode se aprofundar no conjunto de rastreamentos que causam o insight. Para obter mais informações, consulte [Interagir com o console do Analytics](#).

Noções básicas sobre impacto

O AWS X-Ray mede o impacto causado por um problema contínuo como parte da geração de insights e notificações. O impacto é medido de duas maneiras:

- Impacto no [grupo](#) do X-Ray
- Impacto no serviço de causa raiz

Esse impacto é determinado pela porcentagem de solicitações que estão falhando ou causando um erro em um determinado intervalo de tempo. Essa análise de impacto permite que você determine a gravidade e a prioridade do problema com base em seu cenário específico. Esse impacto está disponível como parte da experiência do console, além das notificações de insight.

Desduplicação

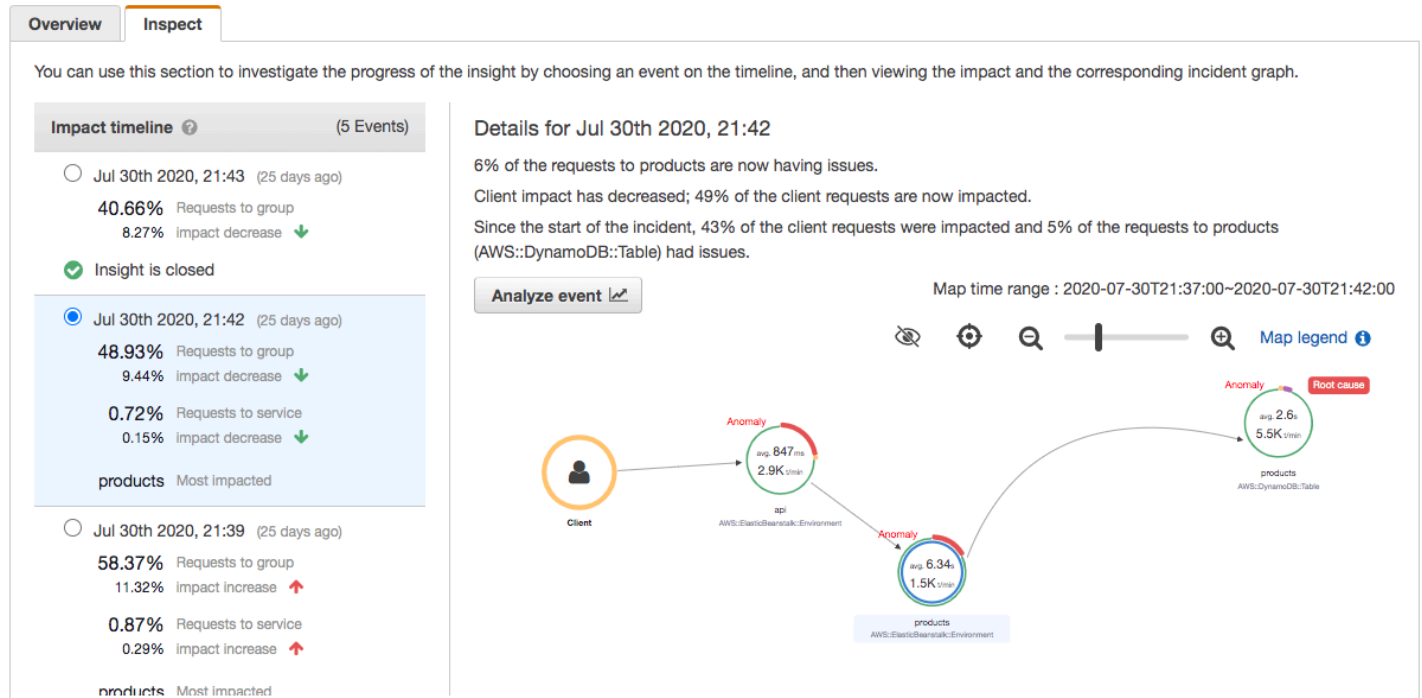
Os insights do AWS X-Ray eliminam a duplicação de problemas em vários microsserviços. O serviço usa a detecção de anomalias para determinar o serviço que é a causa raiz de um problema, determina se outros serviços relacionados estão exibindo um comportamento anômalo devido à mesma causa raiz e registra o resultado como um único insight.

Analisar o andamento de um insight

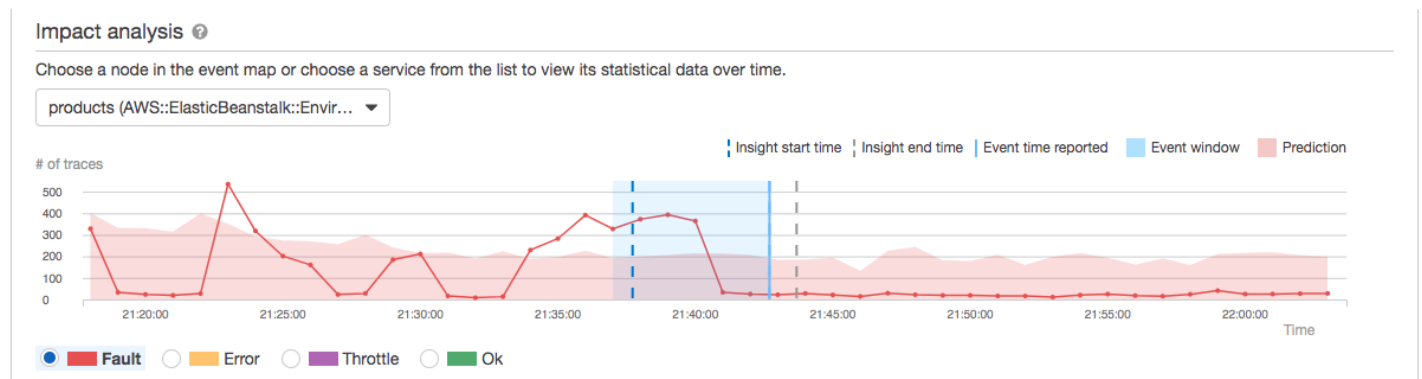
O X-Ray reavalia os insights periodicamente até que sejam resolvidos e registra cada alteração intermediária notável como uma [notificação](#), que pode ser enviada como um evento do Amazon EventBridge. Isso permite que você crie processos e fluxos de trabalho para determinar como o problema mudou ao longo do tempo e tome as medidas apropriadas, como enviar um e-mail ou integrar-se a um sistema de alerta usando o EventBridge.

Você pode analisar os eventos do incidente no Cronograma de impacto na página Inspecionar. Por padrão, o cronograma exibe o serviço mais afetado até que você escolha um serviço diferente.

products (AWS::DynamoDB::Table) of Default group



Para ver um mapa de rastreamento e gráficos de um evento, escolha-o na linha no cronograma de impacto. O mapa de rastreamento mostra os serviços na aplicação que são afetados pelo incidente. Em Análise de impacto, os gráficos mostram cronogramas de falhas para o nó selecionado e os clientes do grupo.



Para analisar mais detalhadamente os rastreamentos envolvidos em um incidente, escolha Analisar evento na página Inspeccionar. Você pode usar a página Análises para refinar a lista de rastreamentos e identificar os usuários afetados. Para obter mais informações, consulte [Interagir com o console do Analytics](#).

Interagir com o console do Analytics

O console do AWS X-Ray Analytics é uma ferramenta interativa de interpretação de dados de rastreamento para entender rapidamente o desempenho do aplicativo e dos serviços subjacentes. O console permite explorar, analisar e visualizar rastreamentos por meio de gráficos interativos de tempo de resposta e de séries temporais.

Ao fazer seleções no console do Analytics, o console cria filtros para refletir o subconjunto selecionado de todos os rastreamentos. É possível refinar o conjunto de dados ativo com filtros cada vez mais granulares clicando nos gráficos e nos painéis de métricas e campos que estão associados ao conjunto de rastreamentos atual.

Tópicos

- [Recursos do console](#)
- [Distribuição do tempo de resposta](#)
- [Atividade de séries temporais](#)
- [Exemplos de fluxo de trabalho](#)
- [Observar as falhas no gráfico de serviço](#)
- [Identificar picos de tempo de resposta](#)
- [Visualizar todos os rastreamentos marcados com um código de status](#)
- [Visualizar todos os itens de um subgrupo de rastreamentos associados a um usuário](#)
- [Compare dois conjuntos de rastreamentos com critérios diferentes](#)
- [Identificar um rastreamento de interesse e visualizar os detalhes](#)

Recursos do console

O console do X-Ray Analytics usa os principais recursos a seguir para agrupamento, filtragem, comparação e quantificação de dados de rastreamento.

Atributos

Recurso	Descrição
Grupos	O grupo selecionado inicial é Default. Para alterar o grupo recuperado, selecione um grupo diferente no menu à direita da barra

Recurso	Descrição
	de pesquisa da expressão de filtro principal. Para saber mais sobre grupos, consulte Usar expressões de filtro com grupos .
Rastreamentos recuperados	Por padrão, o console do Analytics gera gráficos com base em todos os rastreamentos no grupo selecionado. Rastreamentos recuperados representam todos os rastreamentos em seu conjunto de trabalho. Você pode encontrar a contagem de rastreamentos neste bloco. As expressões de filtro aplicadas à barra de pesquisa principal refinam e atualizam os rastreamentos recuperados.
Show in charts/Hide from charts (Mostrar nos gráficos/Ocultar dos gráficos)	Um botão para comparar o grupo ativo com os rastreamentos recuperados. Para comparar os dados relacionados ao grupo com qualquer filtro ativo, escolha Show in charts (Mostrar nos gráficos). Para remover essa visão dos gráficos, escolha Hide from charts (Ocultar dos gráficos).
Conjunto de rastreamentos filtrados A	Por meio de interações com os gráficos e tabelas, aplique filtros para criar os critérios para o conjunto de rastreamentos filtrados A. À medida que os filtros são aplicados, o número de rastreamentos aplicáveis e a porcentagem de rastreamentos do total que são recuperados são calculados neste bloco. Os filtros são preenchidos como tags no bloco Filtered trace set A (Conjunto de rastreamentos filtrados A) e também podem ser removidos do bloco.

Recurso	Descrição
Refinar	Esta função atualiza o conjunto de rastreamentos recuperados com base nos filtros aplicados ao conjunto de rastreamentos A. O refinamento do conjunto de rastreamentos recuperados atualiza o conjunto de trabalho de todos os rastreamentos recuperados com base nos filtros para o conjunto de rastreamentos A. O conjunto de trabalho de rastreamentos recuperados é um subconjunto de amostragem de todos os rastreamentos no grupo.
Conjunto de rastreamentos filtrados B	Quando criado, o conjunto de rastreamentos filtrados B é uma cópia do conjunto de rastreamentos filtrados A. Para comparar os dois conjuntos de rastreamentos, faça novas seleções de filtro que serão aplicadas ao conjunto de rastreamentos B, enquanto o conjunto de rastreamentos A permanece fixo. À medida que os filtros são aplicados, o número de rastreamentos aplicáveis e a porcentagem de rastreamentos do total recuperado são calculados dentro desse bloco. Os filtros são preenchidos como tags no bloco Filtered trace set B (Conjunto de rastreamentos filtrados B) e também podem ser removidos do bloco.

Recurso	Descrição
Response time root cause entity paths (Caminhos da entidade de causa raiz do tempo de resposta)	Uma tabela de caminhos de entidades registrados. O X-Ray determina qual caminho no rastreamento é a causa mais provável do tempo de resposta. O formato indica uma hierarquia de entidades encontradas, terminando em uma causa raiz do tempo de resposta. Use essas linhas para filtrar falhas de tempo de resposta recorrentes. Para obter mais informações sobre como personalizar um filtro de causa raiz e obter dados por meio da API, consulte Recuperar e refinar a análise da causa raiz .
Delta (◆)	Uma coluna que é incluída nas tabelas de métricas quando ambos os conjuntos de rastreamentos A e B estão ativos. A coluna Delta calcula a diferença na porcentagem de rastreamentos entre o conjunto de rastreamentos A e o conjunto de rastreamentos B.

Distribuição do tempo de resposta

O console do X-Ray Analytics gera dois gráficos primários para ajudar você a visualizar rastreamentos: Distribuição do tempo de resposta e Atividade de séries temporais. Esta seção e a seguinte fornecem exemplos de cada uma delas e explicam os conceitos básicos de como ler os gráficos.

Veja a seguir as cores associadas ao gráfico de linha de tempo de resposta (o gráfico de série temporal usa o mesmo esquema de cores):

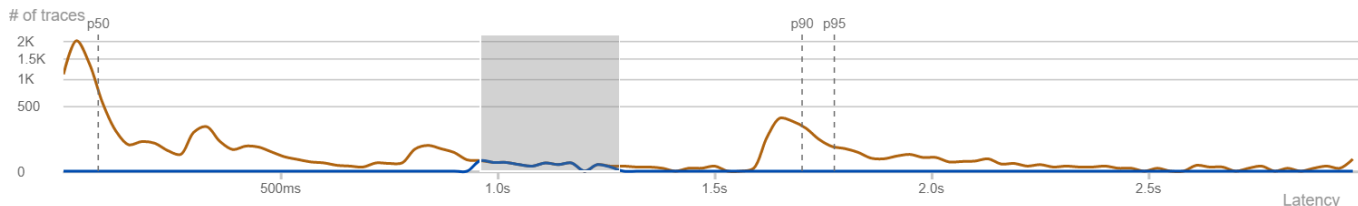
- Todos os rastreamentos no grupo: cinza
- Rastreamentos recuperados: laranja
- Conjunto de rastreamentos filtrados A: verde
- Conjunto de rastreamentos filtrados B: azul

Example Exemplo: distribuição do tempo de resposta

A distribuição do tempo de resposta é um gráfico que mostra o número de rastreamentos com um determinado tempo de resposta. Clique e arraste para fazer seleções dentro da distribuição do tempo de resposta. Isso seleciona e cria um filtro no conjunto de rastreamentos de trabalho nomeado `responseTime` para todos os rastreamentos dentro de um tempo de resposta específico.

Response time distribution

Click and drag to filter the traces by response time.



Atividade de séries temporais

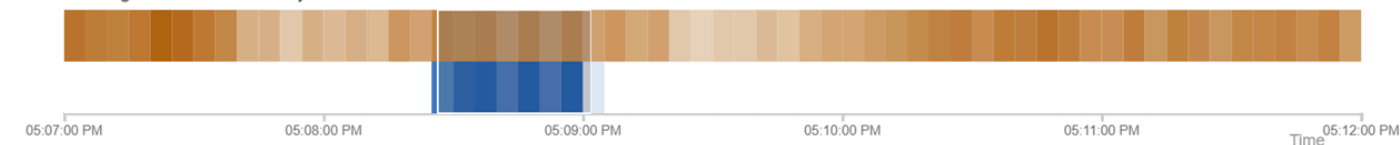
O gráfico de atividade das séries temporais mostra o número de rastreamentos em um determinado período. Os indicadores de cor refletem as cores do gráfico de linhas da distribuição do tempo de resposta. Quanto mais escuro e mais cheio for o bloco de cores dentro da série de atividades, mais traços serão representados no momento determinado.

Example Exemplo: atividade de séries temporais

Clique e arraste para fazer seleções no gráfico de atividades da séries temporais. Isso seleciona e cria um filtro chamado `timerange` no conjunto de rastreamentos de trabalho para todos os rastreamentos dentro de um intervalo de tempo específico.

Time series activity

Click and drag to filter the traces by time.



Exemplos de fluxo de trabalho

Os exemplos a seguir mostram casos de uso comuns do console do X-Ray Analytics. Cada exemplo demonstra uma função-chave da experiência do console. Como um grupo, os exemplos seguem um fluxo de trabalho básico de solução de problemas. As etapas explicam como identificar primeiro os nós não íntegros e, depois, como interagir com o console do Analytics para gerar automaticamente

consultas comparativas. Depois de restringir o escopo por meio de consultas, você finalmente analisará os detalhes dos traços de interesse para determinar o que está prejudicando a integridade do seu serviço.

Observar as falhas no gráfico de serviço

O mapa de rastreamento indica a integridade de cada nó destacando por cores com base no índice de chamadas bem-sucedidas para erros e falhas. Quando você vê uma porcentagem de vermelho no seu nó, ele sinaliza uma falha. Use o console do X-Ray Analytics para investigar.

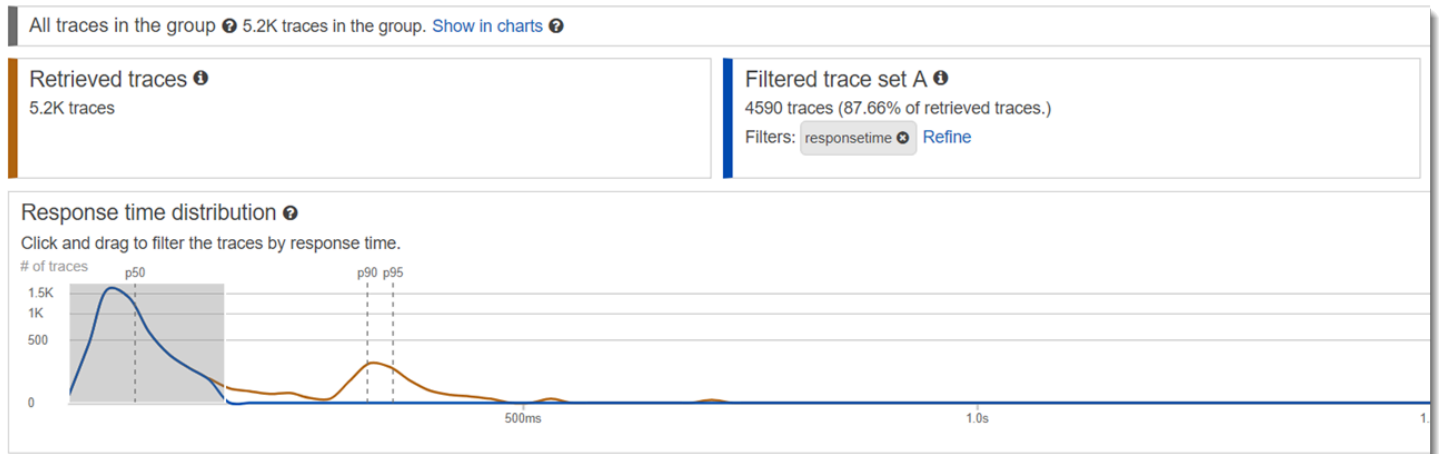
Para obter mais informações sobre como ler o mapa de rastreamento, consulte [Exibir o mapa de rastreamento](#).



Identificar picos de tempo de resposta

Usando a distribuição do tempo de resposta, você pode observar picos no tempo de resposta. Ao selecionar o pico no tempo de resposta, as tabelas abaixo dos gráficos serão atualizadas para expor todas as métricas associadas, como códigos de status.

Quando você clica e arrasta, o X-Ray seleciona e cria um filtro. Ele será exibido em uma sombra cinza na parte superior das linhas em gráficos. Agora você pode arrastar esse destaque para a esquerda e para a direita ao longo da distribuição para atualizar sua seleção.



Visualizar todos os rastreamentos marcados com um código de status

Você pode analisar rastreamentos no pico selecionado usando as métricas tabelas abaixo dos gráficos. Clicando em uma linha na tabela HTTP STATUS CODE (CÓDIGO DE STATUS HTTP), você cria automaticamente um filtro no conjunto de dados de trabalho. Por exemplo, você pode ver todos os rastreamentos do código de status 500. Isso cria uma tag de filtro no bloco do conjunto de rastreamentos chamado `http.status`.

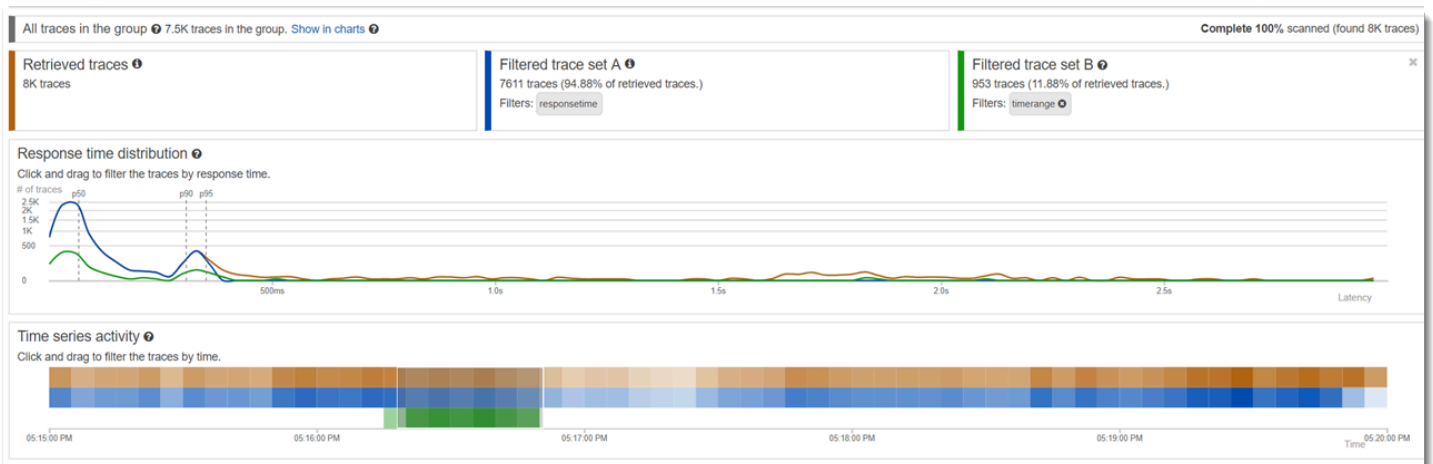
Visualizar todos os itens de um subgrupo de rastreamentos associados a um usuário

Analise o conjunto de erros com base no usuário, URL, causa raiz do tempo de resposta ou outros atributos predefinidos. Por exemplo, para filtrar o conjunto de rastreamentos adicionalmente com um código de status 500, selecione uma linha na tabela USERS (USUÁRIOS). Isso resulta em duas tags de filtro no bloco do conjunto de rastreamentos: `http.status`, conforme designado anteriormente, e `user`.

Compare dois conjuntos de rastreamentos com critérios diferentes

Compare vários usuários e suas solicitações POST para encontrar outras discrepâncias e correlações. Aplique o primeiro conjunto de filtros. Eles são definidos por uma linha azul na distribuição do tempo de resposta. Depois selecione Compare (Comparar). Inicialmente, isso cria uma cópia dos filtros no conjunto de rastreamentos A.

Para continuar, defina um novo conjunto de filtros para aplicar ao conjunto de rastreamentos B. Esse segundo conjunto é representado por uma linha verde. O exemplo a seguir mostra diferentes linhas de acordo com o esquema de cor verde e azul.



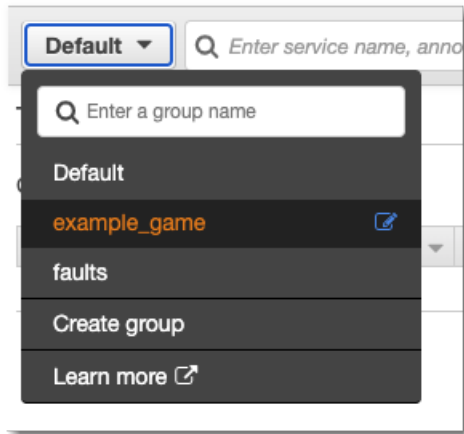
Identificar um rastreamento de interesse e visualizar os detalhes

À medida que você restringe seu escopo usando os filtros de console, a lista de rastreamentos abaixo das tabelas de métricas se torna mais significativa. A tabela da lista de rastreamentos combina informações sobre URL, USER (USUÁRIO) e STATUS CODE (CÓDIGO DE STATUS) em uma exibição. Para obter mais informações, selecione uma linha na tabela para abrir a página de detalhes de rastreamento e visualizar a linha do tempo e dados brutos.

Configurar grupos

Os grupos são uma coleção de rastreamentos definidos por uma expressão de filtro. Você pode usar grupos para gerar gráficos de serviço adicionais e fornecer métricas do Amazon CloudWatch. Você pode usar o console do AWS X-Ray ou a API do X-Ray para criar e gerenciar grupos para seus serviços. Este tópico descreve como criar e gerenciar grupos usando o console do X-Ray. Para obter informações sobre como gerenciar grupos usando a API do X-Ray, consulte [Grupos](#).

Você pode criar grupos de rastreamentos para mapas de rastreamento, rastreamentos ou análises. Ao criar um grupo, ele fica disponível como um filtro no menu suspenso do grupo em todas as três páginas: Mapa de rastreamento, Rastreamentos e Análises.



Os grupos são identificados pelo nome ou pelo nome do recurso da Amazon (ARN) e contêm uma expressão de filtro. O serviço compara os rastreamentos de entrada com a expressão e os armazena adequadamente. Para obter mais informações sobre como criar uma expressão de filtro, consulte [Usar expressões de filtro](#).

Atualizar a expressão de filtro de um grupo não altera os dados que já estão registrados. A atualização se aplica somente aos rastreamentos subsequentes. Isso pode resultar em um gráfico mesclado das expressões novas e antigas. Para evitar isso, exclua o grupo atual e crie outro.

Note

Os grupos são faturados pelo número de rastreamentos recuperados que correspondem à expressão de filtro. Para obter mais informações, consulte [Preços do AWS X-Ray](#).

Tópicos

- [Criar um grupo](#)
- [Aplicar um grupo](#)
- [Editar um grupo](#)
- [Clonar um grupo](#)
- [Excluir um grupo](#)
- [Visualizar métricas de grupo no Amazon CloudWatch](#)

Criar um grupo

Note

Agora você pode configurar as regras de amostragem do X-Ray no console do Amazon CloudWatch. Você também pode continuar usando o console do X-Ray.

CloudWatch console

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Configurações no painel de navegação à esquerda.
3. Escolha Visualizar configurações em Grupos na seção Rastreamentos do X-Ray.
4. Escolha Criar grupo acima da lista de grupos.
5. Na página Criar grupo, digite um nome para o grupo. O nome de grupo pode ter no máximo 32 caracteres, mas somente caracteres alfanuméricos e hifens. Os nomes de grupo diferenciam letras maiúsculas de minúsculas.
6. Insira uma expressão de filtro. Para obter mais informações sobre como criar uma expressão de filtro, consulte [Usar expressões de filtro](#). No exemplo a seguir, o grupo filtra os rastreamentos de falhas do serviço `api.example.com` e as solicitações ao serviço em que o tempo de resposta foi maior ou igual a cinco segundos.

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

7. Em Insights, habilite ou desabilite o acesso a insights para o grupo. Para obter mais informações sobre insights, consulte [Usar os insights do X-Ray](#).

☒ Enable insights

☐ Enable notifications

Deliver insight events using Amazon EventBridge.

8. Em Tags, escolha Adicionar nova tag para inserir uma chave de tag e, opcionalmente, um valor de tag. Continue adicionando outras tags conforme desejado. As chaves de tag devem ser exclusivas. Para excluir uma tag, escolha Remover abaixo de cada tag. Para obter mais informações sobre tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

Key	Value - optional
Q Enter key	Q Enter value
Remove	

9. Escolha Criar grupo.

X-Ray console

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. Abra a página Criar grupo na página Grupos no painel de navegação esquerdo ou no menu de grupos em uma das seguintes páginas: Mapa de rastreamento, Rastreamentos e Análises.
3. Na página Criar grupo, digite um nome para o grupo. O nome de grupo pode ter no máximo 32 caracteres, mas somente caracteres alfanuméricos e hifens. Os nomes de grupo diferenciam letras maiúsculas de minúsculas.
4. Insira uma expressão de filtro. Para obter mais informações sobre como criar uma expressão de filtro, consulte [Usar expressões de filtro](#). No exemplo a seguir, o grupo filtra os rastreamentos de falhas do serviço `api.example.com` e as solicitações ao serviço em que o tempo de resposta foi maior ou igual a cinco segundos.

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

5. Em Insights, habilite ou desabilite o acesso a insights para o grupo. Para obter mais informações sobre insights, consulte [Usar os insights do X-Ray](#).

Enable Insights ☒

Enable Notifications ☒ Deliver insight events using Amazon EventBridge. Learn more about Data Protection in EventBridge. [Learn more](#) 

6. Em Tags, insira uma chave de tag e, opcionalmente, um valor de tag. Ao adicionar uma tag, uma nova linha aparece para você adicionar outra tag. As chaves de tag devem ser exclusivas. Para excluir uma tag, escolha X no final da linha da tag. Para obter mais informações sobre tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

application	game	✕
stage	prod	✕
Key	Value (optional)	✕

7. Escolha Criar grupo.

Aplicar um grupo

CloudWatch console

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Abra uma das páginas a seguir no painel de navegação em Rastreamentos do X-Ray:
 - Mapa de rastreamento
 - Rastreamentos
3. Insira um nome de grupo no filtro Filtrar por grupo do X-Ray. Os dados mostrados na página são alterados para corresponder à expressão de filtro definida no grupo.

X-Ray console

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. Abra uma das seguintes páginas no painel de navegação:
 - Mapa de rastreamento
 - Rastreamentos
 - Analytics
3. No menu de grupos, escolha o grupo em que você criou em [the section called “Criar um grupo”](#). Os dados mostrados na página são alterados para corresponder à expressão de filtro definida no grupo.

Editar um grupo

CloudWatch console

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Configurações no painel de navegação à esquerda.
3. Escolha Visualizar configurações em Grupos na seção Rastreamentos do X-Ray.
4. Escolha um grupo na seção Grupos e selecione Editar.
5. Embora não seja possível renomear um grupo, você pode atualizar a expressão de filtro. Para obter mais informações sobre como criar uma expressão de filtro, consulte [Usar expressões de filtro](#). No exemplo a seguir, o grupo filtra rastreamentos de falha do serviço `api.example.com` em que o endereço do URL da solicitação contém `example/game` e o tempo de resposta para solicitações foi maior ou igual a cinco segundos.

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

6. Em Insights, habilite ou desabilite o acesso a insights para o grupo. Para obter mais informações sobre insights, consulte [Usar os insights do X-Ray](#).

☒ Enable insights

☐ Enable notifications

Deliver insight events using Amazon EventBridge.

7. Em Tags, escolha Adicionar nova tag para inserir uma chave de tag e, opcionalmente, um valor de tag. Continue adicionando outras tags conforme desejado. As chaves de tag devem ser exclusivas. Para excluir uma tag, escolha Remover abaixo de cada tag. Para obter mais informações sobre tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

Key	Value - optional
Q Enter key	Q Enter value
Remove	

8. Quando terminar de atualizar o grupo, escolha Atualizar grupo.

X-Ray console

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. Realize uma das seguintes ações para abrir a página Editar grupo:
 - a. Na página Grupos, escolha o nome de um grupo para editá-lo.
 - b. No menu de grupos em uma das páginas a seguir, aponte para um grupo e escolha Editar.
 - Mapa de rastreamento
 - Rastreamentos
 - Analytics
3. Embora não seja possível renomear um grupo, você pode atualizar a expressão de filtro. Para obter mais informações sobre como criar uma expressão de filtro, consulte [Usar expressões de filtro](#). No exemplo a seguir, o grupo filtra rastreamentos de falha do serviço `api.example.com` em que o endereço do URL da solicitação contém `example/game` e o tempo de resposta para solicitações foi maior ou igual a cinco segundos.

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

4. Em Insights, habilite ou desabilite os insights e as notificações de insights para o grupo. Para obter mais informações sobre insights, consulte [Usar os insights do X-Ray](#).

Enable Insights ☒

Enable Notifications ☒ Deliver insight events using Amazon EventBridge. Learn more about Data Protection in EventBridge. [Learn more](#)

5. Em Tags, edite chaves e valores da tag. As chaves de tag devem ser exclusivas. Os valores das tags são opcionais; você pode excluir valores se quiser. Para excluir uma tag, escolha X no final da linha da tag. Para obter mais informações sobre tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

application	game	X
stage	prod	X
Key	Value (optional)	X

6. Quando terminar de atualizar o grupo, escolha Atualizar grupo.

Clonar um grupo

A clonagem de um grupo cria um grupo com a expressão de filtro e as tags de um grupo existente. Quando você clona um grupo, o novo tem o mesmo nome do grupo do qual ele foi clonado, com `-clone` anexado ao nome.

CloudWatch console

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Configurações no painel de navegação à esquerda.
3. Escolha Visualizar configurações em Grupos na seção Rastreamentos do X-Ray.
4. Escolha um grupo na seção Grupos e selecione Clonar.
5. Na página Criar grupo, o nome do grupo é `group-name-clone`. Opcionalmente, insira um novo nome para o grupo. O nome de grupo pode ter no máximo 32 caracteres, mas somente caracteres alfanuméricos e hifens. Os nomes de grupo diferenciam letras maiúsculas de minúsculas.
6. Você pode manter a expressão de filtro do grupo ou, opcionalmente, inserir uma nova expressão de filtro. Para obter mais informações sobre como criar uma expressão de filtro, consulte [Usar expressões de filtro](#). No exemplo a seguir, o grupo filtra os rastreamentos de falhas do serviço `api.example.com` e as solicitações ao serviço em que o tempo de resposta foi maior ou igual a cinco segundos.

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

7. Em Tags, edite chaves e valores da tag, se necessário. As chaves de tag devem ser exclusivas. Os valores das tags são opcionais; você pode excluir valores se quiser. Para excluir uma tag, escolha X no final da linha da tag. Para obter mais informações sobre tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).
8. Escolha Criar grupo.

X-Ray console

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.

2. Abra a página Grupos no painel de navegação esquerdo e escolha o nome de um grupo que você deseja clonar.
3. No menu Ações, escolha Clonar grupo.
4. Na página Criar grupo, o nome do grupo é `group-name-clone`. Opcionalmente, insira um novo nome para o grupo. O nome de grupo pode ter no máximo 32 caracteres, mas somente caracteres alfanuméricos e hifens. Os nomes de grupo diferenciam letras maiúsculas de minúsculas.
5. Você pode manter a expressão de filtro do grupo ou, opcionalmente, inserir uma nova expressão de filtro. Para obter mais informações sobre como criar uma expressão de filtro, consulte [Usar expressões de filtro](#). No exemplo a seguir, o grupo filtra os rastreamentos de falhas do serviço `api.example.com` e as solicitações ao serviço em que o tempo de resposta foi maior ou igual a cinco segundos.

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

6. Em Tags, edite chaves e valores da tag, se necessário. As chaves de tag devem ser exclusivas. Os valores das tags são opcionais; você pode excluir valores se quiser. Para excluir uma tag, escolha X no final da linha da tag. Para obter mais informações sobre tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).
7. Escolha Criar grupo.

Excluir um grupo

Siga as etapas nesta seção para excluir um grupo. Não é possível excluir o grupo padrão.

CloudWatch console

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Configurações no painel de navegação à esquerda.
3. Escolha Visualizar configurações em Grupos na seção Rastreamentos do X-Ray.
4. Escolha um grupo na seção Grupos e selecione Excluir.
5. Quando for solicitada sua confirmação, escolha Excluir.

X-Ray console

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. Abra a página Grupos no painel de navegação esquerdo e escolha o nome de um grupo que você deseja excluir.
3. No menu Ações, escolha Excluir grupo.
4. Quando for solicitada sua confirmação, escolha Excluir.

Visualizar métricas de grupo no Amazon CloudWatch

Assim que um grupo é criado, os rastreamentos de entrada são verificados em relação à expressão de filtro do grupo, pois são armazenados no serviço X-Ray. As métricas referentes à quantidade de rastreamentos correspondentes a cada critério são publicadas no Amazon CloudWatch a cada minuto. Escolher Exibir métricas na página Editar grupo abre o console do CloudWatch na página Métrica. Para obter mais informações sobre como usar métricas do CloudWatch, consulte [Usar métricas do Amazon CloudWatch](#) no Guia do usuário do Amazon CloudWatch.

CloudWatch console

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Configurações no painel de navegação à esquerda.
3. Escolha Visualizar configurações em Grupos na seção Rastreamentos do X-Ray.
4. Escolha um grupo na seção Grupos e selecione Editar.
5. Na página Editar grupo escolha Exibir métrica.

A página Métricas do console do CloudWatch é aberta em uma nova guia.

X-Ray console

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. Abra a página Grupos no painel de navegação esquerdo e escolha o nome de um grupo do qual você deseja visualizar as métricas.
3. Na página Editar grupo escolha Exibir métrica.

A página Métricas do console do CloudWatch é aberta em uma nova guia.

Configurar regras de amostragem

Você pode usar o AWS X-Ray console para configurar regras de amostragem para seus serviços. O AWS Distro for OpenTelemetry, X-Ray SDK e Serviços da AWS que oferecem suporte ao [rastreamento ativo com configurações de amostragem](#) usam regras de amostragem para determinar quais solicitações devem ser registradas.

Tópicos

- [Configurar regras de amostragem](#)
- [Personalizar regras de amostragem](#)
- [Opções de regras de amostragem](#)
- [Exemplos de regras de amostragem](#)
- [Configurar o serviço para usar regras de amostragem](#)
- [Visualização dos resultados de amostragem](#)
- [Próximas etapas](#)

Configurar regras de amostragem

Você pode configurar a amostragem para os seguintes casos de uso:

- Ponto de entrada do API Gateway: o API Gateway é compatível com amostragem e rastreamento ativo. Para habilitar o rastreamento ativo em um estágio de API, consulte [Compatibilidade com o rastreamento ativo do Amazon API Gateway para o AWS X-Ray](#).
- AWS AppSync— AWS AppSync suporta amostragem e rastreamento ativo. Para ativar o rastreamento ativo nas AWS AppSync solicitações, consulte [Rastreamento com X-Ray AWS](#).
- AWS Step Functions— AWS Step Functions suporta amostragem e rastreamento ativo. Para ativar o rastreamento ativo em máquinas de AWS Step Functions estado, consulte [Rastreamento de X-Ray em Step Functions](#).
- Instrument AWS Distro para OpenTelemetry plataformas computacionais — Ao usar plataformas computacionais como Amazon, EC2 Amazon ECS ou AWS Elastic Beanstalk, a amostragem é suportada quando o aplicativo foi instrumentado com o mais recente AWS Distro for ou X-Ray SDK. OpenTelemetry

Personalizar regras de amostragem

Ao personalizar regras de amostragem, você pode controlar a quantidade de dados registrados. Você também pode modificar o comportamento da amostragem sem modificar ou reimplantar seu código. As regras de amostragem informam ao AWS Distro for OpenTelemetry (ADOT) ou ao X-Ray SDK quantas solicitações devem ser registradas de acordo com um conjunto de critérios. Por padrão, o SDK registra a primeira solicitação a cada segundo e cinco por cento de todas as solicitações adicionais. Uma solicitação por segundo é o reservatório. Isso garante que pelo menos um rastreamento seja registrado a cada segundo à medida que o serviço atende às solicitações. Cinco por cento é a taxa segundo a qual as solicitações adicionais, além do tamanho de reservatório, são amostradas.

Você pode configurar o X-Ray SDK para ler as regras de amostragem de um documento JSON incluído com seu código. No entanto, ao executar várias instâncias do seu serviço, cada instância realiza a amostragem de forma independente. Isso faz com que a porcentagem total de solicitações amostradas aumente, pois os reservatórios de todas as instâncias são efetivamente somados. Além disso, para atualizar as regras de amostragem locais, você deve reimplantar seu código.

Ao definir regras de amostragem no console do X-Ray e [configurar o SDK](#) para ler as regras do serviço do X-Ray, você pode evitar esses problemas. O serviço gerencia o reservatório para cada regra e atribui cotas para cada instância de seu serviço a fim de distribuir o reservatório uniformemente com base no número de instâncias que estão em execução. O limite do reservatório é calculado de acordo com as regras definidas. Como as regras são configuradas no serviço, você pode gerenciá-las sem fazer implantações adicionais.

Note

Ao configurar as regras de amostragem, é fundamental entender que a amostragem por X-Ray é “baseada nos pais”. Isso significa que a decisão de amostragem é tomada apenas uma vez, normalmente pelo primeiro X-Ray-enabled serviço que processa a solicitação (o serviço “Root”).

Se um serviço downstream receber uma solicitação que já tenha uma decisão de amostragem de um pai upstream, ele honrará essa decisão independentemente de qualquer uma de suas próprias regras de amostragem correspondentes.

- Quando as regras se aplicam: Suas regras de amostragem personalizadas só entram em vigor em serviços em que a decisão de amostragem ainda não foi tomada. Isso geralmente se aplica a:

- O ponto de entrada do seu aplicativo (por exemplo, um API Gateway, um Load Balancer ou o primeiro microsserviço instrumentado).
- Processos ou trabalhadores assíncronos que iniciam um novo rastreamento.
- Armadilha comum: se você criar uma regra de amostragem estrita para o “Serviço B”, mas o “Serviço B” sempre for chamado pelo “Serviço A”, sua regra para o Serviço B provavelmente nunca será aplicada porque está simplesmente seguindo a decisão passada pelo Serviço A. Para alterar a amostragem de traços desse fluxo de trabalho, você deve configurar a regra de amostragem para o serviço raiz (Serviço A).

Note

Como o X-Ray usa a abordagem de melhor esforço na aplicação de regras de amostragem, em alguns casos a taxa de amostragem efetiva pode não corresponder exatamente às regras de amostragem configuradas. No entanto, com o tempo, o número de solicitações amostradas deve estar próximo à porcentagem configurada.

Agora você pode configurar as regras de amostragem do X-Ray no CloudWatch console da Amazon.

Para configurar as regras de amostragem no console CloudWatch

1. Faça login no Console de gerenciamento da AWS e abra o CloudWatch console em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Configurações no painel de navegação esquerdo, abaixo de Configuração.
3. Escolha Ver configurações em Regras de amostragem na seção Rastreamento do X-Ray.
4. Para criar uma regra, escolha Criar regra de amostragem.

Para editar uma regra, escolha uma regra e selecione Editar para editá-la.

Para excluir uma regra, escolha uma regra e selecione Excluir, para excluí-la.

Opções de regras de amostragem

As opções a seguir estão disponíveis para cada regra. Valores de string podem usar curingas para corresponder a um caractere único (?) ou zero ou mais caracteres (*).

Opções de regras de amostragem

- Nome da regra (string): um nome exclusivo para a regra.
- Prioridade (inteiro entre 1 e 9.999): a prioridade da regra de amostragem. Os serviços avaliam as regras em ordem decrescente de prioridade e tomam uma decisão de amostragem com a primeira regra correspondente.
- Reservatório (inteiro não negativo): um número fixo de solicitações correspondentes para instrumentar por segundo, antes de aplicar a taxa fixa. O reservatório não é usado diretamente pelos serviços, mas se aplica a todos os serviços usando a regra coletivamente.
- Taxa (número inteiro entre 0 e 100): a porcentagem de solicitações correspondentes para instrumentar, depois que o reservatório está esgotado. Ao configurar uma regra de amostragem no console, escolha uma porcentagem entre 0 e 100. Ao configurar uma regra de amostragem em um SDK de cliente usando um documento JSON, forneça um valor percentual entre 0 e 1.
- Nome do serviço (string): o nome do serviço instrumentado, como ele aparece no mapa de rastreamento.
 - X-Ray SDK: o nome do serviço que você configura no gravador.
 - Amazon API Gateway: *api-name/stage*.
- Tipo do serviço (string): o tipo de serviço, como ele aparece no mapa de rastreamento. Para o X-Ray SDK, defina o tipo de serviço aplicando o plug-in apropriado:
 - `AWS::ElasticBeanstalk::Environment`— Um AWS Elastic Beanstalk ambiente (plugin).
 - `AWS::EC2::Instance`— Uma EC2 instância da Amazon (plugin).
 - `AWS::ECS::Container`: um contêiner do Amazon ECS (plug-in).
 - `AWS::EKS::Container`— Um contêiner Amazon EKS (plugin).
 - `AWS::APIGateway::Stage`: um estágio do Amazon API Gateway.
 - `AWS::AppSync::GraphQLAPI` — Uma solicitação de AWS AppSync API.
 - `AWS::StepFunctions::StateMachine`— Uma máquina AWS Step Functions estatal.
- Host (string): o nome de host do cabeçalho de host HTTP.
- Método HTTP (string): o método da solicitação HTTP.
- Caminho do URL (string): o caminho URL da solicitação.
 - X-Ray SDK: a porção do caminho URL da solicitação HTTP.
- ARN do recurso (string) — O ARN do AWS recurso que está executando o serviço.
 - X-Ray SDK: sem suporte. O SDK só pode usar regras com o Resource ARN (ARN do recurso) definido como `*`.

- Amazon API Gateway: o ARN do estágio.
- (Opcional) Atributos (chave e valor): atributos de segmento que são conhecidos quando a decisão de amostragem é feita.
 - X-Ray SDK: sem suporte. O SDK ignora as regras que especificam atributos.
 - Amazon API Gateway: cabeçalhos da solicitação HTTP original.
- (Opcional) SamplingRateBoost(objeto) — Controla o comportamento de aumento de amostragem baseado em anomalias.
 - MaxRate (número inteiro entre 0 e 100) — Taxa máxima de amostragem (0,0—1,0) O raio-X pode aumentar para durante anomalias
 - CooldownWindowMinutes (número inteiro maior que 0) — Janela de tempo (minutos) na qual somente um impulso pode ser acionado, evitando aumentos contínuos

Exemplos de regras de amostragem

Example Exemplo: regra padrão sem reservatório e com taxa baixa

Você pode modificar o reservatório e a taxa da regra padrão. A regra padrão se aplica às solicitações que não correspondem a nenhuma outra regra.

- Reservatório: **0**
- Taxa: **5 (0.05)** se configurada usando um documento JSON)

Example Exemplo: regra de depuração para rastrear todas as solicitações para uma rota problemática

Uma regra de alta prioridade aplicada temporariamente para depuração.

- Nome da regra: **DEBUG – history updates**
- Prioridade: **1**
- Reservatório: **1**
- Taxa: **100 (1)** se configurada usando um documento JSON)
- Nome de serviço: **Scorekeep**
- Tipo de serviço: *****
- Host: *****

- Método HTTP: **PUT**
- Caminho URL: **/history/***
- ARN do recurso: *

Example — Taxa mínima mais alta para POSTs

- Nome da regra: **POST minimum**
- Prioridade: **100**
- Reservatório: **10**
- Taxa: **10** (**.1** se configurada usando um documento JSON)
- Nome de serviço: *
- Tipo de serviço: *
- Host: *
- Método HTTP: **POST**
- Caminho URL: *
- ARN do recurso: *

Example Ativar o aumento impulsionado por anomalias

Configure uma regra que acione um aumento de amostragem para até 50% durante anomalias, com um tempo de desaquecimento de 10 minutos.

- Nome da regra: **MyAdaptiveRule**
- Prioridade: **100**
- Reservatório: **1**
- FixedRate: **0.0510**
- Nome de serviço: *
- Tipo de serviço: *
- Host: *
- Método HTTP: **POST**
- Caminho URL: *
- maxRate: **0.5**

- `cooldownWindowMinutes`: **10**

Configurar o serviço para usar regras de amostragem

O AWS Distro for OpenTelemetry (ADOT) e o X-Ray SDK exigem configuração adicional para usar as regras de amostragem que você configura no console. Consulte o tópico de configuração referente à sua linguagem para obter detalhes sobre como configurar uma estratégia de amostragem:

- Java: [Usando a amostragem remota X-Ray](#) com ADOT Java
- Go: [Configurando a amostragem](#) com o ADOT Go
- Node.js: [Usando a amostragem remota X-Ray com ADOT](#) JavaScript
- Python: Usando o [X-Ray Remote Sampling](#) com ADOT Python
- Ruby: [Regras de amostragem](#)
- .NET: [Usando o X-Ray Remote Sampling](#) com ADOT.NET

Para API Gateway, consulte [Compatibilidade com o rastreamento ativo do Amazon API Gateway para o AWS X-Ray](#).

Visualização dos resultados de amostragem

A página Amostragem do console do X-Ray mostra informações detalhadas sobre como seus serviços usam cada regra de amostragem.

A coluna Trend (Tendência) mostra como a regra foi usada nos últimos minutos. Cada coluna mostra estatísticas para uma janela de 10 segundos.

Estatísticas da amostragem

- Total de regras correspondidas: o número de solicitações que correspondem a essa regra. Esse número não inclui solicitações que poderiam ter correspondido essa regra, mas que primeiro encontraram uma regra correspondente de prioridade mais alta.
- Total de amostra o número de solicitações registradas.
- Amostradas com taxa fixa: o número de solicitações amostradas aplicando a taxa fixa da regra.
- Amostra com reservatório limite: o número de solicitações amostradas usando uma cota atribuída pelo X-Ray.

- **Emprestado do reservatório:** o número de solicitações amostradas por empréstimo do reservatório. Na primeira vez em que um serviço corresponde a uma solicitação de uma regra, ele não recebe uma cota do X-Ray. No entanto, se o reservatório for pelo menos 1, o serviço tomará emprestado um rastreamento por segundo até que o X-Ray atribua uma cota.

Para obter mais informações sobre as estatísticas de amostragem e como usar os serviços de regras de amostragem, consulte [Usar regras de amostragem com a API do X-Ray](#).

Próximas etapas

Você pode usar a API do X-Ray para gerenciar as regras de amostragem. Com a API, você pode criar e atualizar regras de forma programática, em uma programação ou em resposta a alarmes ou notificações. Consulte [Definir configurações de amostragem, grupos e criptografia com a API do AWS X-Ray](#) para obter instruções e ver mais exemplos de regras.

O AWS Distro for OpenTelemetry, o X-Ray SDK Serviços da AWS também usa a API X-Ray para ler regras de amostragem, relatar resultados de amostragem e obter alvos de amostragem. Os serviços devem controlar a frequência com que eles aplicam cada regra, avaliam as regras com base na prioridade e tomam emprestado do reservatório quando uma solicitação corresponde a uma regra para a qual o X-Ray ainda não tiver atribuído uma cota de serviço. Para obter mais detalhes sobre como um serviço usa a API de amostragem, consulte [Usar regras de amostragem com a API do X-Ray](#).

Quando o AWS Distro for OpenTelemetry ou o X-Ray SDK chamam a amostragem APIs, eles usam o CloudWatch agente como proxy. Se você já usa a porta TCP 2000, pode configurar o agente para executar o proxy em uma porta diferente. Consulte o [guia de instalação do CloudWatch agente](#) para obter detalhes.

Configurar amostragem adaptável

A falta de rastreamentos críticos durante picos de anomalias pode dificultar a análise da causa raiz. No entanto, a manutenção de altas taxas de amostragem é cara. A amostragem adaptativa do X-Ray proporciona visibilidade completa das anomalias e controla os custos durante as operações normais. Com a amostragem adaptativa, você define uma taxa máxima de amostragem, e o X-Ray se ajusta automaticamente dentro desse limite. O X-Ray calcula o aumento mínimo necessário para capturar rastreamentos de erros. Se sua taxa de base capturar dados suficientes, nenhum aumento ocorrerá. Você só paga pela amostragem extra quando necessário.

Benefícios do uso da amostragem adaptável:

- Visibilidade completa do incidente: obtenha rastreamentos completos durante os incidentes sem intervenção manual. O X-Ray ajusta automaticamente as taxas de amostragem para capturar rastreamentos de erros e, depois, retorna às taxas normais.
- Visibilidade da causa raiz: confira sempre a origem dos problemas. O X-Ray captura dados de erros críticos mesmo quando a amostragem completa do traço não é acionada.
- Otimize os custos: breves aumentos de amostragem (até 1 minuto) e períodos de espera automático evitam a sobreamostragem. Você só paga pelos dados necessários para diagnosticar problemas.

Tópicos

- [SDKs e plataformas compatíveis](#)
- [Escolha sua abordagem de amostragem adaptável](#)
- [Configuração do SDK local](#)

SDKs e plataformas compatíveis

SDK compatível: a amostragem adaptável requer a versão mais recente do SDK do ADOT.

Linguagem compatível: Java (versão [v2.11.5](#) ou superior)

Sua aplicação deve ser instrumentada com o SDK do ADOT compatível e executada com o agente do Amazon CloudWatch ou o coletor do OpenTelemetry.

Por exemplo, o Amazon EC2, o Amazon ECS e o Amazon EKS são plataformas comuns nas quais o AWS Application Signals fornece orientações para ativar o SDK do ADOT e o agente do Amazon CloudWatch.

Escolha sua abordagem de amostragem adaptável

A amostragem adaptável é compatível com duas abordagens: Aumento de amostragem e Captura de extensões de anomalias. Elas podem ser aplicadas de maneira independente ou podem ser combinadas.

Aumento de amostragem

O aumento adaptável de amostragem se baseia em regras de amostragem e funciona com o modelo existente de amostragem com base no principal do X-Ray. Amostragem baseada em principais

significa que as decisões de amostragem são tomadas no serviço raiz, e o indicador de amostragem é passado downstream para todos os serviços na cadeia de chamadas.

- Aumento baseado em regras: o aumento está sempre vinculado a uma regra específica de amostragem do X-Ray. Cada regra pode definir sua própria taxa máxima de aumento e comportamento de espera.
- Amostragem baseada em principal: as decisões de amostragem são tomadas no serviço raiz, e o indicador de amostragem é passado downstream para todos os serviços na cadeia de chamadas.
- Orientado por anomalias: o X-Ray depende do SDK para relatar estatísticas de anomalias. Quando o X-Ray detecta anomalias, como erros ou alta latência, ele usa essas estatísticas para calcular uma taxa de aumento apropriada (até o máximo configurado).

Relatórios de anomalias

Cada serviço de aplicativo na cadeia de chamadas pode emitir estatísticas de anomalias por meio do SDK necessário:

- Serviço raiz: deve ser executado em um SDK e uma plataforma compatíveis para permitir o aumento da amostragem. Se o serviço raiz não for compatível, nenhum aumento ocorrerá.
- Serviços downstream: os serviços downstream relatam apenas anomalias; eles não podem tomar decisões de amostragem. Quando um serviço downstream está executando um SDK compatível, as anomalias detectadas podem acionar um aumento na amostragem. Quando um serviço downstream não é compatível (por exemplo, executando um SDK antigo), anomalias nesse serviço não acionarão um aumento. Esses serviços ainda podem propagar o contexto downstream quando seguem a propagação padrão do contexto (como contexto de rastreamento e bagagem do W3C). Isso garante que os SDKs compatíveis em outros serviços downstream possam relatar anomalias que desencadeiam um aumento.

Aumentar o tempo e o escopo

- Atraso do gatilho: um aumento de amostragem pode começar em apenas 10 segundos após o X-Ray detectar uma anomalia.
- Período de aumento: depois que o X-Ray aciona um aumento, ele dura até 1 minuto antes de retornar à taxa de base de amostragem.
- Aumento de espera: após a ocorrência de um aumento, o X-Ray só acionará outro impulso para a mesma regra depois que a janela de espera tiver passado.

Por exemplo, quando você define `cooldown` para 10 minutos, quando um aumento termina, nenhum novo aumento poderá ser acionado até a próxima janela de 10 minutos.

Caso especial: quando você define `cooldown` para 1 minuto, e como o aumento em si pode durar até 1 minuto, os impulsos podem ser acionados de maneira eficaz e contínua se a anomalia persistir.

Note

Use SDKs e plataformas compatíveis para seu serviço raiz. O aumento de amostragem funciona somente com SDKs e plataformas compatíveis. Embora o aumento de amostragem tenha uma alta probabilidade de capturar rastreamentos de anomalias, ele pode não capturar todos os rastreamentos de anomalia.

Aumentar a visibilidade

Quando uma regra de amostragem é configurada com aumento de amostragem adaptável, o X-Ray emite automaticamente métricas fornecidas que permitem monitorar a atividade do aumento.

- Nome da métrica – `SamplingRate`
- Dimensão — `RuleName` (definida como o nome real da regra)

Cada regra com `SamplingRateBoost` habilitado publicará sua taxa de amostragem efetiva, incluindo a taxa de base e quaisquer aumentos temporários. Isso permite a você:

- Monitorar quando os aumentos são acionados
- Monitorar a taxa de amostragem efetiva para cada regra
- Correlacionar os aumentos com anomalias do aplicativo (como picos de erros ou eventos de latência)

Você pode visualizar essas métricas no Amazon CloudWatch Metrics, em `AWS/X-Ray` namespace. O valor métrico é um número de ponto flutuante entre 0 e 1, representando a taxa de amostragem efetiva.

Configurar o aumento de amostragem usando regras de amostragem do X-Ray

Você pode habilitar a amostragem adaptável diretamente nas regras existentes de amostragem do X-Ray adicionando um novo campo `SamplingRateBoost`. Para obter mais informações, consulte [Personalizar regras de amostragem](#). Essa é uma maneira centralizada de permitir a amostragem adaptável sem modificar o código do aplicativo ou aplicar a implantação dele. Quando você habilita a amostragem adaptativa, o X-Ray aumenta automaticamente a amostragem durante anomalias, como picos de erro ou valores discrepantes de latência, mantendo as taxas de amostragem dentro do limite máximo configurado. `SamplingRateBoost` pode ser aplicado a qualquer regra de amostragem personalizada, exceto à regra de amostragem `Default`.

O campo `SamplingRateBoost` define o limite superior e o comportamento da amostragem baseada em anomalias.

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

`MaxRate` define a taxa máxima de amostragem que o X-Ray aplicará ao detectar anomalias. O intervalo de valores é de `0.0` a `1.0`. Por exemplo, `"MaxRate": 0.25` permite que a amostragem aumente em até 25% das solicitações durante uma janela de anomalia. O X-Ray determina a taxa apropriada entre a sua linha de base e a máxima, dependendo da atividade da anomalia.

`CooldownWindowMinutes` define a janela de tempo (em minutos) na qual apenas um aumento na taxa de amostragem pode ser acionado. Depois que um aumento ocorre, nenhum aumento adicional é permitido até a próxima janela. O tipo de valor é inteiro (minutos).

Exemplo de regra com amostragem adaptável

```
{
  "RuleName": "MyAdaptiveRule",
  "Priority": 1,
  "ReservoirSize": 1,
  "FixedRate": 0.05,
  "ServiceName": "*",
  "ServiceType": "*",
  "Host": "*",
  "HTTPMethod": "*",
```

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

Neste exemplo, a amostragem de linha de base é 5% (`FixedRate: 0.05`). Durante anomalias, o X-Ray pode aumentar a amostragem em até 25% (`MaxRate: 0.25`). Aumente apenas uma vez a cada 10 minutos.

Configuração da condição de anomalia

Quando nenhuma configuração de condição de anomalia é fornecida, o SDK do ADOT usa códigos de erro HTTP 5xx como condição de anomalia padrão para acionar o aumento da amostragem.

Você também pode ajustar as condições de anomalia localmente no SDK do ADOT compatível usando variáveis de ambiente. Para obter mais informações, consulte [Configuração do SDK local](#).

Captura de extensões de anomalias

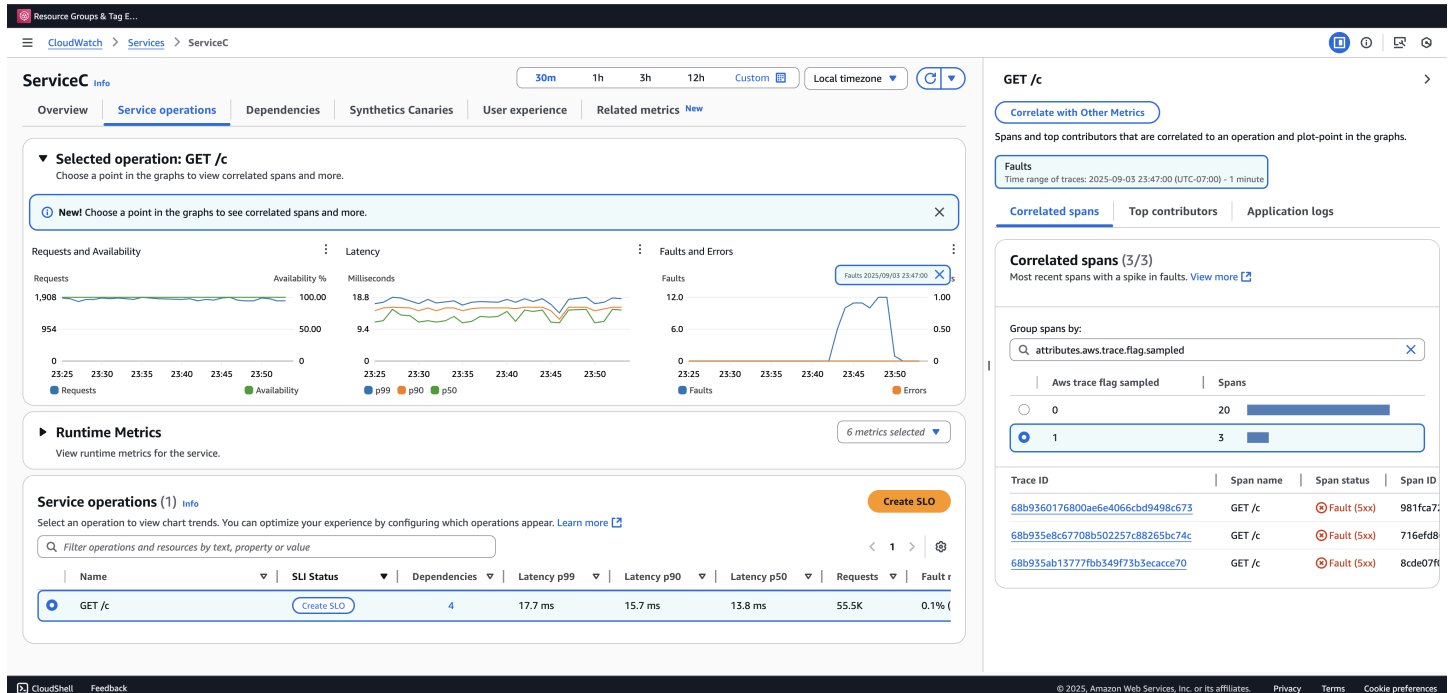
A captura da extensão de anomalias garante que as extensões críticas que representam anomalias sejam sempre registradas, mesmo que o rastreamento completo não seja amostrado. Esse recurso complementa o aumento da amostragem, concentrando-se na captura da anomalia em si, em vez de aumentar a amostragem para futuros rastreamentos.

Quando o SDK do ADOT detecta uma anomalia, ele emite essa extensão imediatamente, independentemente da decisão de amostragem. Como o SDK só emite extensões relacionadas à anomalia, esses rastreamentos são parciais, não transações completas de ponta a ponta.

Depois que o SDK do ADOT detecta uma extensão de anomalia, ele tenta emitir o maior número possível de extensões do mesmo rastreamento. Todas as extensões emitidas nesse recurso são marcadas com o atributo, `aws.trace.flag.sampled = 0`. Isso permite distinguir facilmente traços parciais (captura de anomalias) de traços completos (amostragem normal) na pesquisa e análise de transações.

É recomendável integrar a [Pesquisa de transações](#) para visualizar e consultar rastreamentos parciais. O exemplo a seguir mostra uma página de serviço no console do [Application Signals](#). O ServiceC é configurado com captura de extensões de anomalias e faz parte de uma cadeia de

chamadas em que o aumento da amostragem se aplica. Essas configurações geram rastreamentos completos e parciais. Você pode usar o atributo `aws.trace.flag.sampled` para distinguir entre os tipos de rastreamento.



A captura de extensões de anomalias só pode ser habilitada ou personalizada por meio do [Configuração do SDK local](#).

Configuração do SDK local

Você pode configurar recursos de amostragem adaptável no SDK do ADOT fornecendo uma configuração YAML por meio de uma variável de ambiente. A configuração local fornece controle refinado sobre condições e limites de anomalias.

Isso é necessário para a captura de extensões de anomalias e opcional para personalizar as condições de aumento de amostragem. Veja a seguir um exemplo de configuração:

```
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
```

Confira as definições de campo abaixo:

- **version:** versão do esquema do arquivo de configuração
- **anomalyConditions:** define as condições sob as quais as anomalias são detectadas e como elas são usadas
 - **errorCodeRegex:** expressão regular que define quais códigos de status HTTP são considerados anomalias
 - **operations:** lista de operações ou endpoints aos quais a condição se aplica
 - **highLatencyMs:** limite de latência (em milissegundos) acima do qual as extensões são tratadas como anomalias
 - **usage:** define a qual recurso a condição se aplica:
 - **both:** aplica-se ao aumento de amostragem e à captura de extensões de anomalias (padrão se o uso não for especificado)
 - **sampling-boost:** usado somente para acionar aumentos de amostragem
 - **anomaly-span-capture:** usado somente para captura de extensões de anomalias
- **anomalyCaptureLimit:** define os limites para a emissão de rastreamentos com extensões de anomalias.

anomalyTracesPerSecond: número máximo de rastreamentos com extensões de anomalias capturadas por segundo para evitar um volume excessivo de extensões (o valor padrão será 1 se **anomalyCaptureLimit** não estiver presente).

Note

- **AnomalyConditions** substitui a condição de anomalia padrão para o aumento de amostragem (HTTP 5xx). Se você quiser reter a condição padrão ao usar a configuração local, deverá incluí-la explicitamente em qualquer item de **AnomalyConditions**.

- Para cada item `anomalyConditions`:
 - Quando o campo `operations` é omitido, a condição se aplica a todas as operações (nível de serviço)
 - Quando o campo `operations` está presente, mas definido como uma lista vazia, a condição se aplica a nenhuma operação, tornando esse item sem operação
 - Quando `errorCodeRegex` e `highLatencyMs` são omitidos, a condição não tem critérios de anomalia a serem avaliados, tornando esse item sem operação
- Relações lógicas:
 - Entre os itens em `anomalyConditions`, o relacionamento é OR.
 - Em um único item, vários campos (por exemplo, `errorCodeRegex` e `highLatencyMs`) são combinados com AND.

Por exemplo:

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

Essa condição significa que o código de status corresponde a 429 ou 5xx E a latência é ≥ 300 ms.

Aplique a configuração local ao SDK do ADOT

Você pode aplicar a configuração local ao SDK do ADOT definindo a variável de ambiente `AWS_XRAY_ADAPTIVE_SAMPLING_CONFIG`. O valor deve ser um documento YAML válido (em linha ou aninhado).

Por exemplo, o Amazon EC2 e o Amazon ECS definem a variável de ambiente de maneira direta:

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


Para o Amazon EKS, defina a variável de ambiente dentro da especificação do pod como YAML aninhado:

```
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
```

Vinculação direta do console

Você pode usar rotas e consultas para vinculação direta em rastreamentos específicos ou exibições filtradas de rastreamentos e mapas de rastreamento.

Páginas do console

- Página de boas-vindas: [xray/home#/welcome](#)
- Conceitos básicos: [xray/home#/getting-started](#)
- Mapa de rastreamento: [xray/home#/service-map](#)
- Rastreamentos: [xray/home#/traces](#)

Rastreamentos

Você pode gerar links para as visualizações de cronograma, bruta e de mapa de rastreamentos individuais.

Linha do tempo do rastreamento: `xray/home#/traces/trace-id`

Dados de rastreamento brutos: `xray/home#/traces/trace-id/raw`

Example Exemplo: dados de rastreamento brutos

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

Expressões de filtro

Link para uma lista filtrada de rastreamentos.

Visualização de rastreamentos filtrados: `xray/home#/traces?filter=filter-expression`

Example Exemplo: expressão de filtro

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

Example Exemplo: expressão de filtro (URL codificado)

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

Para obter mais informações sobre expressões de filtro, consulte [Usar expressões de filtro](#).

Intervalo de tempo

Especifique um período ou hora inicial e final no formato ISO8601. Os intervalos de tempo estão em UTC e podem ser de até seis horas.

Duração: `xray/home#/page?timeRange=range-in-minutes`

Example : mapa de rastreamento para a última hora

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

Horas de início e término: xray/home#/*page*?timeRange=*start~end*

Example Exemplo: intervalo de tempo preciso em segundos

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

Example Exemplo: intervalo de tempo preciso em minutos

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

Região

Especifique uma Região da AWS a ser vinculada a páginas nessa região. Caso não especifique uma região, o console redirecionará você para a região visitada mais recentemente.

Região: xray/home?**region=region**#/*page*

Example : mapa de rastreamento no Oeste dos EUA (Oregon) (us-west-2)

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

Quando você inclui uma região com outros parâmetros de consulta, a consulta da região fica antes do hash e as consultas específicas do X-Ray ficam depois do nome da página.

Example : mapa de rastreamento na última hora no Oeste dos EUA (Oregon) (us-west-2)

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

Combinado

Example Exemplo: rastreamentos recentes com um filtro de duração

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

Resultado

- Página: Rastreamentos
- Período: últimos 15 minutos
- Filtro: duração ≥ 5 E duração ≤ 8

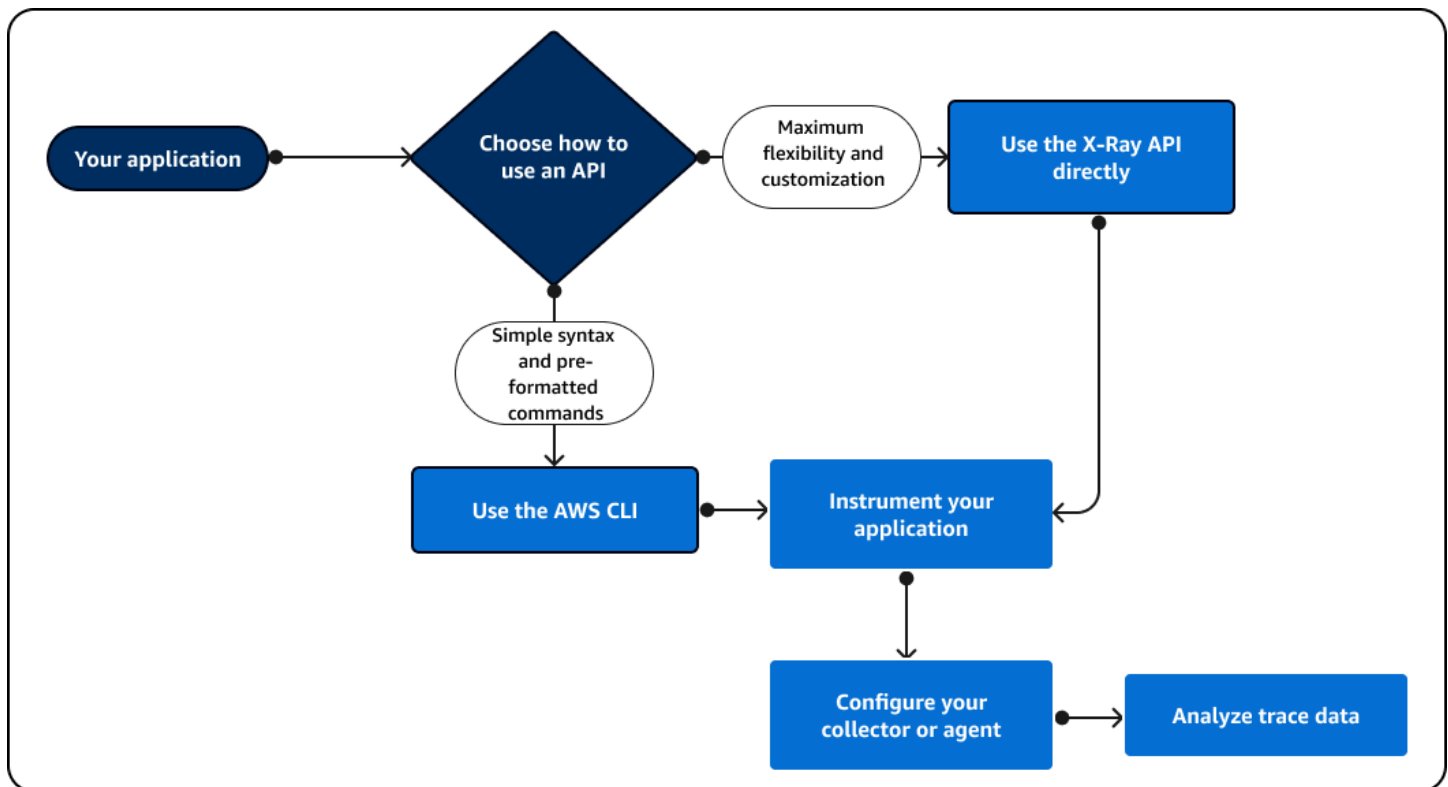
Usar a API do X-Ray

Se o SDK do X-Ray não for compatível com sua linguagem de programação, você poderá usar as APIs do X-Ray diretamente ou o AWS Command Line Interface (AWS CLI) para chamar os comandos da API do X-Ray. Use as orientações a seguir para escolher como interagir com a API:

- Use a AWS CLI para obter uma sintaxe mais simples usando comandos pré-formatados ou com opções dentro de sua solicitação.
- Use a API do X-Ray diretamente para obter o máximo de flexibilidade e personalização das solicitações que você fizer ao X-Ray.

Se você usar a [API do X-Ray](#) diretamente em vez da AWS CLI, será necessário parametrizar sua solicitação no formato de dados correto e talvez configurar a autenticação e o tratamento de erros.

O diagrama a seguir mostra orientações para escolher como interagir com a API do X-Ray:



Use a API do X-Ray para enviar dados de rastreamento diretamente para o X-Ray. A API do X-Ray é compatível com todas as funções disponíveis no SDK do X-Ray, incluindo as seguintes ações comuns:

- [PutTraceSegments](#): faz upload de documentos do segmento para o X-Ray.
- [BatchGetTraces](#): recupera uma lista de rastreamentos em uma lista de IDs de rastreamento. Cada rastreamento recuperado é uma coleção de documentos de segmento de uma única solicitação.
- [GetTraceSummaries](#): recupera IDs e anotações de rastreamentos. Você pode especificar uma `FilterExpression` para recuperar um subconjunto de resumos de rastreamento.
- [GetTraceGraph](#): recupera um gráfico de serviço para um ID de rastreamento específico.
- [GetServiceGraph](#): recupera um documento formatado em JSON que descreve serviços que processam solicitações de entrada e chamam solicitações subsequentes.

Você também pode usar o AWS Command Line Interface (AWS CLI) dentro do código da aplicação para interagir programaticamente com o X-Ray. A AWS CLI é compatível com todas as funções disponíveis no SDK do X-Ray, incluindo aquelas para outros Serviços da AWS. As funções a seguir são versões das operações de API listadas anteriormente com um formato mais simples:

- [put-trace-segments](#): faz upload de documentos do segmento para o X-Ray.

- [batch-get-traces](#): recupera uma lista de rastreamentos em uma lista de IDs de rastreamento. Cada rastreamento recuperado é uma coleção de documentos de segmento de uma única solicitação.
- [get-trace-summaries](#): recupera IDs e anotações de rastreamentos. Você pode especificar uma `FilterExpression` para recuperar um subconjunto de resumos de rastreamento.
- [get-trace-graph](#): recupera um gráfico de serviço para um ID de rastreamento específico.
- [get-service-graph](#): recupera um documento formatado em JSON que descreve serviços que processam solicitações de entrada e chamam solicitações subsequentes.

Para começar, você deve instalar a [AWS CLI](#) para o seu sistema operacional. A AWS oferece suporte aos sistemas operacionais Linux, macOS e Windows. Para obter mais informações sobre a lista de comandos do X-Ray, consulte o [Guia de referência de comandos de AWS CLI para o X-Ray](#).

Tópicos

- [Utilizar a API do AWS X-Ray com a AWS CLI](#)
- [Envio de dados de rastreamento para AWS X-Ray](#)
- [Obter dados do AWS X-Ray](#)
- [Definir configurações de amostragem, grupos e criptografia com a API do AWS X-Ray](#)
- [Usar regras de amostragem com a API do X-Ray](#)
- [AWS X-Ray Documentos de segmento do](#)

Utilizar a API do AWS X-Ray com a AWS CLI

A AWS CLI permite que você acesse o serviço X-Ray diretamente e utilize as mesmas APIs que o console do X-Ray usa para recuperar o gráfico de serviço e os dados de rastreamento não processados. O aplicativo de exemplo inclui scripts que mostram como usar essas APIs com a AWS CLI.

Pré-requisitos

Este tutorial usa o aplicativo de amostra Scorekeep e scripts incluídos para gerar dados de rastreamento e um mapeamento de serviço. Siga as instruções no [tutorial de conceitos básicos](#) para iniciar o aplicativo.

Este tutorial utiliza a AWS CLI para mostrar o uso básico da API do X-Ray. A AWS CLI, [disponível para Windows, Linux e OS-X](#), fornece acesso à linha de comando para as APIs públicas para todos os Serviços da AWS.

 Note

Você deve verificar se a AWS CLI está configurada na mesma região em que o aplicativo de exemplo Scorekeep foi criado.

Os scripts incluídos para testar o aplicativo de exemplo usam cURL para enviar tráfego para a API e jq para analisar a saída. Você pode baixar do executável jq a partir de stedolan.github.io e do executável curl a partir de <https://curl.haxx.se/download.html>. A maioria das instalações Linux e OS X incluem cURL.

Gerar dados de rastreamento

O aplicativo Web continua a gerar tráfego para a API a cada poucos segundos enquanto o jogo está em andamento, mas apenas gera um tipo de solicitação. Use o script `test-api.sh` para executar cenários de ponta a ponta e gerar mais dados diversificados de rastreamento enquanto testa a API.

Para usar o script **test-api.sh**

1. Abra o [console do Elastic Beanstalk](#).
2. Navegue até o [console de gerenciamento](#) do seu ambiente.
3. Copie o URL de ambiente do cabeçalho da página.
4. Abra `bin/test-api.sh` e substitua o valor para API com o URL do seu ambiente.

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

5. Execute o script para gerar tráfego para a API.

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

```
[ "JQVLE0M2", "D67QLPIC", "VF9BM9NC", "0EAA6GK9", "2A705073", "1U2LFTLJ", "HUKIDD70", "BAN1C8FI", "G
["BS8F8LQ", "4MTTSPKP", "4630ETES", "SVEBCL3N", "N7CQ1GHP", "0840NEPD", "EG4BPROQ", "V4BLIDJ3", "9R
```

Usar a API do X-Ray

A AWS CLI fornece comandos para todas as ações de API que o X-Ray fornece, incluindo [GetServiceGraph](#) e [GetTraceSummaries](#). Consulte a [AWS X-Ray Referência da API](#) para mais informações sobre todas as ações suportadas e os tipos de dados que usam.

Example bin/service-graph.sh

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

O script recupera um gráfico de serviço para os últimos 10 minutos.

```
~/eb-java-scorekeep$ ./bin/service-graph.sh | less
{
  "StartTime": 1479068648.0,
  "Services": [
    {
      "StartTime": 1479068648.0,
      "ReferenceId": 0,
      "State": "unknown",
      "EndTime": 1479068651.0,
      "Type": "client",
      "Edges": [
        {
          "StartTime": 1479068648.0,
          "ReferenceId": 1,
          "SummaryStatistics": {
            "ErrorStatistics": {
              "ThrottleCount": 0,
              "TotalCount": 0,
              "OtherCount": 0
            },
            "FaultStatistics": {
              "TotalCount": 0,
              "OtherCount": 0
            },
            "TotalCount": 2,
            "OkCount": 2,
```



```

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

```

Example bin/trace-urls.sh

```

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

```

O script recupera os URLs de rastreamentos gerados entre um e dois minutos atrás.

```

~/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[*]'
```

O script recupera rastreamentos completos gerados entre um e dois minutos atrás.

```
~/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\":\"QAKE0S8DD0LJM245KAOPMA746BVV4KQNS05AEMVJF66Q9ASUAAJG\",
        \"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
        \":\"388GER0C4PCA6D59ED3CTI5EEJVV4KQNS05AEMVJF66Q9ASUAAJG\", \"resource_names\":
        [\"scorekeep-game-xray\"]},\"origin\":\"AWS::DynamoDB::Table\"}"
      }
    ],
    "Id": "1-5828d9f2-a90669393f4343211bc1cf75",
    "Duration": 0.05099987983703613
  }
  ...
]
```

Limpeza

Encerre seu ambiente do Elastic Beanstalk para desativar as instâncias do Amazon EC2, as tabelas do DynamoDB e outros recursos.

Como encerrar o ambiente do Elastic Beanstalk

1. Abra o [console do Elastic Beanstalk](#).

2. Navegue até o [console de gerenciamento](#) do seu ambiente.
3. Escolha Ações.
4. Escolha Terminate Environment.
5. Escolha Encerrar.

Os dados de rastreamento são excluídos automaticamente do X-Ray após 30 dias.

Envio de dados de rastreamento para AWS X-Ray

É possível enviar dados de rastreamento para o X-Ray em forma de documentos segmentados. Um documento segmentado é uma string formatada por JSON que contém informações sobre o trabalho que o aplicativo faz diante de uma solicitação. O aplicativo pode registrar dados sobre o trabalho que ele mesmo faz em segmentos ou o trabalho que usa serviços e recursos de downstream em subsegmentos.

Os segmentos registram informações sobre o trabalho que o aplicativo faz. Pelo menos um segmento registra o tempo gasto em uma tarefa, um nome e duas IDs. A ID de rastreamento rastreia a solicitação à medida que ela percorre os serviços. A ID de segmento rastreia o trabalho feito para a solicitação por um único serviço.

Example Segmento completo mínimo

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

Quando uma solicitação é recebida, é possível enviar um segmento em andamento como um espaço reservado até a solicitação estar concluída.

Example Segmento em andamento

```
{
  "name" : "Scorekeep",
  "id" : "70de5b6f19ff9a0b",
```

```
{
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  "in_progress": true
}
```

É possível enviar segmentos ao X-Ray diretamente, com [PutTraceSegments](#) ou , [por meio do daemon do X-Ray](#).

A maioria dos aplicativos chama outros serviços ou acessa recursos com o SDK do AWS. Registre informações sobre chamadas subsequentes nos subsegmentos. O X-Ray usa subsegmentos para identificar serviços subsequentes que não enviam segmentos e criam entradas para eles no gráfico de serviço.

Um subsegmento pode ser incorporado em um documento de segmento completo ou enviado separadamente. Envie subsegmentos separadamente para rastrear chamadas subsequentes de forma assíncrona para solicitações de longa duração ou para evitar exceder o tamanho máximo do documento de segmentos (64 kB).

Example Subsegmento

Um subsegmento tem um type de subsegment e uma parent_id que identifica o segmento pai.

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

Para obter mais informações sobre os campos e valores que você pode incluir em segmentos e subsegmentos, consulte [AWS X-Ray Documentos de segmento do](#) .

Seções

- [Gerar IDs de rastreamento](#)
- [Uso de PutTraceSegments](#)
- [Enviar documentos de segmentos para o daemon do X-Ray](#)

Gerar IDs de rastreamento

Para enviar dados ao X-Ray, é necessário gerar um ID de rastreamento exclusivo para cada solicitação.

Formato do ID de rastreamento do X-Ray

Um `trace_id` do X-Ray consiste em três números separados por hifens. Por exemplo, `1-58406520-a006649127e371903a2de979`. Isso inclui:

- O número da versão, que é 1.
- A hora da solicitação original, em horário epoch Unix, com 8 dígitos hexadecimais.

Por exemplo, 10h no dia 1º de dezembro de 2016 PST equivale a `1480615200` segundos em horário epoch ou a `58406520` em dígitos hexadecimais.

- Um identificador globalmente exclusivo de 96 bits para o rastreamento com 24 dígitos hexadecimais.

Note

O X-Ray agora aceita IDs de rastreamento criados usando o OpenTelemetry ou qualquer outro framework que esteja em conformidade com a [especificação W3C Trace Context](#). Um ID de rastreamento do W3C deve ser formatado conforme o ID de rastreamento do X-Ray ao ser enviado ao X-Ray. Por exemplo, o ID de rastreamento `4efaaf4d1e8720b39541901950019ee5` do W3C deve ser formatado como `1-4efaaf4d-1e8720b39541901950019ee5` quando enviado ao X-Ray. Os IDs de rastreamento do X-Ray incluem o carimbo de data e hora da solicitação original no horário epoch Unix, mas isso não é necessário ao enviar IDs de rastreamento do W3C no formato do X-Ray.

Você pode gravar um script para gerar IDs de rastreamento do X-Ray para testes. Veja dois exemplos a seguir.

Python

```
import time
import os
```

```
import binascii

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

Bash

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

Consulte o aplicativo de exemplo Scorekeep para scripts que criam IDs de rastreamento e enviam segmentos ao daemon do X-Ray.

- Python – [xray_start.py](#)
- Bash: [xray_start.sh](#)

Uso de PutTraceSegments

É possível fazer upload de documentos segmentados com a API [PutTraceSegments](#). A API tem um único parâmetro, `TraceSegmentDocuments`, que utiliza uma lista de documentos segmentados JSON.

Com a AWS CLI, use o comando `aws xray put-trace-segments` para enviar documentos de segmentos diretamente para o X-Ray.

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

Note

O Processador de Comandos do Windows e o Windows PowerShell têm requisitos diferentes para aspas e aspas de escape em strings JSON. Consulte [Colocação de strings](#) no AWS CLI Guia do usuário para detalhes.

A saída lista todos os segmentos que falharam no processamento. Por exemplo, caso a data na ID de rastreamento seja muito antiga, você vê um erro como o erro a seguir.

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

É possível passar vários documentos segmentados simultaneamente, separados por espaços.

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

Enviar documentos de segmentos para o daemon do X-Ray

Em vez de enviar documentos de segmentos para a API do X-Ray, você pode enviar segmentos e subsegmentos para o daemon do X-Ray, que os armazena em buffer e os carrega em lote na API do X-Ray. O X-Ray SDK envia documentos segmentados ao daemon para evitar fazer chamadas para a AWS diretamente.

Note

Consulte [Executar o daemon do X-Ray localmente](#) para obter instruções sobre como executar o daemon.

Envie o segmento em JSON pela porta UDP 2000, acrescido do cabeçalho do daemon, `{"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"}
```

No Linux, é possível enviar documentos segmentados para o daemon de um terminal Bash. Salve o cabeçalho e o documento segmentado em um arquivo de texto e o encapsule em `/dev/udp` com `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"}
```

Verifique o [log do daemon](#) para confirmar se ele enviou o segmento ao X-Ray.

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

Obter dados do AWS X-Ray

AWS X-RayO processa dados de rastreamento que você envia a ele para gerar rastreamentos completos, resumos de rastreamentos e gráficos de serviço em JSON. Você pode recuperar os dados gerados diretamente da API com a AWS CLI.

Seções

- [Recuperar o gráfico de serviço](#)
- [Recuperar o gráfico de serviços por grupo](#)
- [Recuperar rastreamentos](#)
- [Recuperar e refinar a análise da causa raiz](#)

Recuperar o gráfico de serviço

Você pode usar a API do [GetServiceGraph](#) para recuperar o gráfico de serviço do JSON. A API requer um horário de início e término, que você pode calcular a partir de um terminal Linux com o comando `date`.

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

`date +%s` imprime uma data em segundos. Use esse número como um horário de término e subtraia o tempo dele para obter um horário de início.

Example Script para recuperar um gráfico de serviço dos últimos 10 minutos

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

O exemplo a seguir mostra um gráfico de serviço com quatro nós, incluindo um nó do cliente, uma instância do EC2, uma tabela do DynamoDB e um tópico do Amazon SNS.

Example Saída do GetServiceGraph

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

```

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

```

```

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

```

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

```

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

```

```

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

```

```

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

```

Recuperar o gráfico de serviços por grupo

Para chamar um gráfico de serviço baseado no conteúdo de um grupo, inclua um `groupName` ou `groupARN`. O exemplo a seguir mostra uma chamada de gráfico de serviço para um grupo chamado `Example1`.

Example Script para recuperar um gráfico de serviço por nome para o grupo `Example1`

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

Recuperar rastreamentos

Você pode usar a API do [GetTraceSummaries](#) para obter uma lista dos resumos de rastreamentos. Os resumos de rastreamentos incluem informações que você pode usar para identificar rastreamentos cujo download total deseja fazer, incluindo anotações, informações de solicitação e resposta e IDs.

Há duas bandeiras `TimeRangeType` disponíveis ao chamar `aws xray get-trace-summaries`:

- **TraceID**: a pesquisa `GetTraceSummaries` padrão usa o tempo do `TraceID` e retorna os rastreamentos iniciados dentro do intervalo `[start_time, end_time)` computado. Esse intervalo de timestamps é calculado com base na codificação do timestamp dentro do `TraceID` ou pode ser definido manualmente.
- **Horário do evento**: para pesquisar eventos à medida que eles acontecem ao longo do tempo, o AWS X-Ray permite pesquisar rastreamentos usando registros de data e hora dos eventos. A hora do evento retorna traços ativos durante o intervalo `[start_time, end_time)`, independentemente de quando o rastreamento começou.

Use o comando `aws xray get-trace-summaries` para obter uma lista dos resumos de rastreamentos. Os comandos a seguir obtêm uma lista de resumos de rastreamento entre 1 e 2 minutos no passado usando o tempo de `TraceID` padrão.

Example Script para obter resumos de rastreamentos

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

Example Saída do `GetTraceSummaries`

```
{
  "TraceSummaries": [
    {
      "HasError": false,
      "Http": {
        "HttpStatus": 200,
        "ClientIp": "205.255.255.183",
        "HttpURL": "http://scorekeep.elasticbeanstalk.com/api/session",
        "UserAgent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36",
        "HttpMethod": "POST"
      }
    }
  ]
}
```



```

    },
    "Users": [],
    "HasFault": false,
    "Annotations": {},
    "ResponseTime": 0.084,
    "Duration": 0.084,
    "Id": "1-59602606-a43a1ac52fc7ee0eea12a82c",
    "HasThrottle": false
  },
  {
    "HasError": false,
    "Http": {
      "HttpStatus": 200,
      "ClientIp": "205.255.255.183",
      "HttpURL": "http://scorekeep.elasticbeanstalk.com/api/user",
      "UserAgent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36",
      "HttpMethod": "POST"
    },
    "Users": [
      {
        "UserName": "5M388M1E"
      }
    ],
    "HasFault": false,
    "Annotations": {
      "UserID": [
        {
          "AnnotationValue": {
            "StringValue": "5M388M1E"
          }
        }
      ],
      "Name": [
        {
          "AnnotationValue": {
            "StringValue": "01a"
          }
        }
      ]
    },
    "ResponseTime": 3.232,
    "Duration": 3.232,
    "Id": "1-59602603-23fc5b688855d396af79b496",

```

```

        "HasThrottle": false
    },
    ],
    "ApproximateTime": 1499473304.0,
    "TracesProcessedCount": 2
}

```

Use o ID de rastreamento da saída para recuperar um rastreamento completo com a API do [BatchGetTraces](#).

Example Comando BatchGetTraces

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

Example Saída do BatchGetTraces

```
{
  "Traces": [
    {
      "Duration": 3.232,
      "Segments": [
        {
          "Document": "{\"id\":\"1fb07842d944e714\",\"name\":\
\"random-name\",\"start_time\":1.499473411677E9,\"end_time\":1.499473414572E9,\
\"parent_id\":\"0c544c1b1bbff948\",\"http\":{\"response\":{\"status\":200}},\
\"aws\":{\"request_id\":\"ac086670-6373-11e7-a174-f31b3397f190\"},\"trace_id\":\
\"1-59602603-23fc5b688855d396af79b496\",\"origin\":\"AWS::Lambda\",\"resource_arn\":\
\"arn:aws:lambda:us-west-2:123456789012:function:random-name\"}\",
          "Id": "1fb07842d944e714"
        },
        {
          "Document": "{\"id\":\"194fcc8747581230\",\"name\":\"Scorekeep
\",\"start_time\":1.499473411562E9,\"end_time\":1.499473414794E9,\"http\":{\"request
\":{\"url\":\"http://scorekeep.elasticbeanstalk.com/api/user\",\"method\":\"POST\",
\"user_agent\":\"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/59.0.3071.115 Safari/537.36\",\"client_ip\":\"205.251.233.183\"},
\"response\":{\"status\":200}},\"aws\":{\"elastic_beanstalk\":{\"version_label\":\"app-
abb9-170708_002045\",\"deployment_id\":406,\"environment_name\":\"scorekeep-dev\"},
\"ec2\":{\"availability_zone\":\"us-west-2c\",\"instance_id\":\"i-0cd9e448944061b4a
\"},\"xray\":{\"sdk_version\":\"1.1.2\",\"sdk\":\"X-Ray for Java\"}},\"service
\":{\"},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"user\":\"5M388M1E
\",\"origin\":\"AWS::ElasticBeanstalk::Environment\",\"subsegments\": [{\"id\":
\"0c544c1b1bbff948\",\"name\":\"Lambda\",\"start_time\":1.499473411629E9,\"end_time
```

```

\":1.499473414572E9,\"http\":{\"response\":{\"status\":200,\"content_length\":14}},
\"aws\":{\"log_type\":\"None\",\"status_code\":200,\"function_name\":\"random-name
\", \"invocation_type\":\"RequestResponse\", \"operation\":\"Invoke\", \"request_id
\": \"ac086670-6373-11e7-a174-f31b3397f190\", \"resource_names\": [\"random-name\"]},
\"namespace\":\"aws\"}, {\"id\":\"071684f2e555e571\", \"name\":\"## UserModel.saveUser
\", \"start_time\":1.499473414581E9, \"end_time\":1.499473414769E9, \"metadata\":{\"debug
\": {\"test\":\"Metadata string from UserModel.saveUser\"}}, \"subsegments\": [{\"id\":
\"4cd3f10b76c624b4\", \"name\":\"DynamoDB\", \"start_time\":1.49947341469E9, \"end_time
\":1.499473414769E9, \"http\":{\"response\":{\"status\":200,\"content_length\":57}},
\"aws\":{\"table_name\":\"scorekeep-user\", \"operation\":\"UpdateItem\", \"request_id
\": \"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738B0B4KQNS05AEMVJF66Q9\", \"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\":\"01a\"}}, {\"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-6B829NT9V509\", \"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\":
\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738B0B4KQNS05AEMVJF66Q9\", \"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-6B829NT9V509\\"},
\\"trace_id\\":\\"1-59602603-23fc5b688855d396af79b496\\",\\"origin\\":\\"AWS::SNS\\"}",
      "Id": "1ee3c4a523f89ca5"
    }
  ],
  "Id": "1-59602603-23fc5b688855d396af79b496"
}
  ],
  "UnprocessedTraceIds": []
}

```

O rastreamento completo inclui um documento para cada segmento, compilado a partir de todos os documentos de segmento recebidos com a mesma ID de rastreamento. Esses documentos não representam os dados como eles foram enviados para o X-Ray pelo aplicativo. Em vez disso, eles representam os documentos processados e gerados pelo serviço X-Ray. O X-Ray cria o documento de rastreamento completo compilando os documentos de segmentos enviados pelo aplicativo e removendo os dados que não estão em conformidade com o [esquema do documento de segmento](#).

O X-Ray também cria segmentos inferidos para chamadas subsequentes para os serviços que não enviam segmentos por conta própria. Por exemplo, quando você faz chamadas para o DynamoDB com um cliente instrumentado, o X-Ray SDK grava um subsegmento com detalhes sobre a chamada de acordo com o ponto de vista dele. No entanto, o DynamoDB não envia um segmento correspondente. O X-Ray usa as informações no subsegmento para criar um segmento inferido para representar o recurso DynamoDB no mapa de rastreamento e o adiciona ao documento de rastreamento.

Para obter vários rastreamentos da API, você precisa de uma lista de IDs de rastreamento, que pode ser extraída da saída de um `get-trace-summaries` com uma [AWS CLI consulta](#). Redirecione a lista para a entrada `batch-get-traces` para obter rastreamentos completos de um período específico.

Example Script para obter rastreamentos completos de um período de um minuto

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

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

Recuperar e refinar a análise da causa raiz

Ao gerar um resumo de rastreamentos com a [API do GetTraceSummaries](#), os resumos de rastreamentos parciais podem ser reutilizados em seu formato JSON para criar uma expressão de filtro refinada com base nas causas raiz. Veja os exemplos abaixo para obter uma demonstração das etapas de refinamento.

Example Exemplo de saída do GetTraceSummaries: seção de causa raiz do tempo de resposta

```
{  
  "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
    }
  ]
}
]
}

```

Ao editar e omitir a saída acima, esse JSON pode se tornar um filtro para entidades de causa raiz correspondentes. Para cada campo presente no JSON, qualquer correspondência de candidato deve ser exata ou o rastreamento não será retornado. Os campos removidos se tornam valores curinga, um formato compatível com a estrutura de consulta da expressão de filtro.

Example Causa raiz de tempo de resposta reformatada

```

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

```

Este JSON é usado como parte de uma expressão de filtro por meio de uma chamada para `rootcause.json = #[{ }]`. Consulte o capítulo [Filter Expressions \(Filtrar Expressões\)](#) para obter mais detalhes sobre a consulta com expressões de filtro.

Example Exemplo de filtro JSON

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

Definir configurações de amostragem, grupos e criptografia com a API do AWS X-Ray

AWS X-Ray fornece APIs para configurar [regras de amostragem](#), regras de grupo e [configurações de criptografia](#).

Seções

- [Configurações de criptografia](#)
- [Regras de amostragem](#)
- [Grupos](#)

Configurações de criptografia

Use [PutEncryptionConfig](#) para especificar uma chave do AWS Key Management Service (AWS KMS) a ser usada para criptografia.

Note

O X-Ray não aceita chaves do KMS assimétricas.

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

Para o ID de chave, você pode usar um alias (conforme mostrado no exemplo), um ID de chave ou um nome de recurso da Amazon (ARN).

Use [GetEncryptionConfig](#) para obter a configuração atual. Quando o X-Ray termina de aplicar as configurações, o status é alterado de UPDATING para ACTIVE.

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

Para deixar de usar uma chave do KMS e usar a criptografia padrão, defina o tipo de criptografia como NONE.

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

Regras de amostragem

Você pode gerenciar as [regras de amostragem](#) em sua conta com a API do X-Ray. Para obter mais informações sobre como adicionar e gerenciar tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

Obtenha todas as regras de amostragem com [GetSamplingRules](#).

```
$ aws xray get-sampling-rules
{
  "SamplingRuleRecords": [
    {
      "SamplingRule": {
        "RuleName": "Default",
```



```

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

```

A regra padrão se aplica a todas as solicitações que não correspondem a nenhuma outra regra. Ela é a regra de prioridade mais baixa e não pode ser excluída. No entanto, você pode alterar a taxa e o tamanho do reservatório com a [UpdateSamplingRule](#).

Example Entrada de API para [UpdateSamplingRule](#): 10000-default.json

```

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

```

O exemplo a seguir usa o arquivo anterior como entrada para alterar a regra padrão para um por cento, sem reservatório. As tags são opcionais. Se você optar por adicionar tags, uma chave de tag será necessária e os valores da tag serão opcionais. Para remover tags de uma regra de amostragem, use [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
},
```

Crie regras de amostragem adicionais com a [CreateSamplingRule](#). Ao criar uma regra, a maioria dos campos da regra são obrigatórios. O exemplo a seguir cria duas regras. Esta primeira regra define uma taxa de base para o aplicativo de exemplo do Scorekeep. Ela faz a correspondência de todas as solicitações atendidas pela API que não correspondem a uma regra de prioridade mais alta.

Example Entrada de API para [UpdateSamplingRule](#): 9000-base-scorekeep.json

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

A segunda regra também se aplica ao Scorekeep, mas tem uma prioridade mais alta e é mais específica. Essa regra define uma taxa de amostragem muito baixa para solicitações de sondagem. Essas são solicitações GET feitas pelo cliente a cada poucos segundos para verificar a existência de alterações no estado do jogo.

Example Entrada de API para [UpdateSamplingRule](#): 5000-polling-scorekeep.json

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

As tags são opcionais. Se você optar por adicionar tags, uma chave de tag será necessária e os valores da tag serão opcionais.

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

```

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

```

Para excluir uma regra de amostragem, use a [DeleteSamplingRule](#).

```

$ aws xray delete-sampling-rule --rule-name polling-scorekeep
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "polling-scorekeep",
            "RuleARN": "arn:aws:xray:us-east-1:123456789012:sampling-rule/polling-
scorekeep",
            "ResourceARN": "*",
            "Priority": 5000,

```

```

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

```

Grupos

Você pode usar a API do X-Ray para gerenciar grupos em sua conta. Os grupos são uma coleção de rastreamentos definidos por uma expressão de filtro. Você pode usar grupos para gerar gráficos de serviço adicionais e fornecer métricas do Amazon CloudWatch. Consulte [Obter dados do AWS X-Ray](#) para obter mais detalhes sobre como trabalhar com métricas e gráficos de serviço por meio da API do X-Ray. Para obter mais informações sobre grupos, consulte [Configurar grupos](#). Para obter mais informações sobre como adicionar e gerenciar tags, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

Criar um grupo com `CreateGroup`. As tags são opcionais. Se você optar por adicionar tags, uma chave de tag será necessária e os valores da tag serão opcionais.

```

$ 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}"
}

```

Obtenha todos os grupos existentes `GetGroups`.

```

$ aws xray get-groups
{

```

```

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

```

Atualizar um grupo com `UpdateGroup`. As tags são opcionais. Se você optar por adicionar tags, uma chave de tag será necessária e os valores da tag serão opcionais. Para remover tags de uma regra de um grupo, use [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}"
}

```

Excluir um grupo com `DeleteGroup`.

```

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

```

Usar regras de amostragem com a API do X-Ray

O AWS X-Ray SDK usa a API do X-Ray para obter regras de amostragem, informar os resultados da amostragem e obter cotas. Você pode usar essas APIs para compreender melhor como as regras

de amostragem funcionam ou para implementar a amostragem em uma linguagem que não seja compatível com o X-Ray SDK.

Comece obtendo todas as regras de amostragem com [GetSamplingRules](#).

```
$ aws xray get-sampling-rules
{
  "SamplingRuleRecords": [
    {
      "SamplingRule": {
        "RuleName": "Default",
        "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/Default",
        "ResourceARN": "*",
        "Priority": 10000,
        "FixedRate": 0.01,
        "ReservoirSize": 0,
        "ServiceName": "*",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1,
        "Attributes": {}
      },
      "CreatedAt": 0.0,
      "ModifiedAt": 1530558121.0
    },
    {
      "SamplingRule": {
        "RuleName": "base-scorekeep",
        "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/base-scorekeep",
        "ResourceARN": "*",
        "Priority": 9000,
        "FixedRate": 0.1,
        "ReservoirSize": 2,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1,
        "Attributes": {}
      },
      "CreatedAt": 1530573954.0,
```

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

A saída inclui a regra padrão e regras personalizadas. Consulte [Regras de amostragem](#) se você ainda não tiver criado regras de amostragem.

Avalie as regras em relação às solicitações de entrada em ordem crescente de prioridade. Quando houver uma correspondência de regras, use a taxa fixa e o tamanho de reservatório para tomar uma decisão de amostragem. Registre as solicitações amostradas e ignore (para fins de rastreamento) as solicitações não amostradas. Pare de avaliar as regras quando uma decisão de amostragem for tomada.

O tamanho de um reservatório de regras é o número de destino de rastreamentos a serem registrados por segundo antes do aplicativo da taxa fixa. O reservatório é aplicado em todos os serviços cumulativamente. Portanto, você não pode usá-lo diretamente. No entanto, se ele for diferente de zero, você poderá tomar emprestado um rastreamento por segundo do reservatório até que o X-Ray atribua uma cota. Antes de receber uma cota, registre a primeira solicitação a cada segundo, e aplique a taxa fixa nas solicitações adicionais. A taxa fixa é um valor decimal entre 0 e 1,00 (100%).

O exemplo a seguir mostra uma chamada para [GetSamplingTargets](#) com detalhes sobre as decisões de amostragem tomadas nos últimos 10 segundos.

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

A resposta do X-Ray inclui uma cota para ser usada no lugar de empréstimos do reservatório. Neste exemplo, o serviço tomou emprestado 10 rastreamentos do reservatório em 10 segundos e aplicou a taxa fixa de 10% para as outras 100 solicitações, o que resulta em um total de 20 solicitações amostradas. A cota é válida por cinco minutos (indicada pela vida útil) ou até que uma nova cota seja atribuída. O X-Ray também pode atribuir um intervalo de emissão de relatórios maior do que o padrão, embora isso não ocorra aqui.

Note

A resposta do X-Ray pode não incluir uma cota na primeira vez em que você chamá-lo. Continue fazendo empréstimos do reservatório até que você receba uma cota.

Os outros dois campos na resposta podem indicar problemas com a entrada. Compare a `LastRuleModification` com a data da última vez em que você chamou as [GetSamplingRules](#). Se ela for mais recente, obtenha uma nova cópia das regras. `UnprocessedStatistics` pode incluir erros que indicam que uma regra foi excluída, que o documento de estatísticas na entrada era muito antigo ou erros de permissões.

AWS X-Ray Documentos de segmento do

Um segmento de rastreamento é uma representação JSON de uma solicitação que seu aplicativo atende. Um segmento de rastreamento registra informações sobre a solicitação original, informações sobre o trabalho que o aplicativo faz localmente e subsegmentos com informações sobre chamadas subsequentes que o aplicativo faz a APIs HTTP, bancos de dados SQL e recursos da AWS.

Um documento de segmentos transmite informações sobre um segmento ao X-Ray. Um documento de segmentos pode ser até 64 kB e conter um segmento inteiro com subsegmentos, um fragmento de um segmento que indica que uma solicitação está em andamento ou um único subsegmento enviado separadamente. Você pode enviar documentos de segmentos diretamente ao X-Ray usando a API [PutTraceSegments](#).

O X-Ray compila e processa documentos de segmentos para gerar resumos de rastreamento e rastreamentos completos, os quais podem ser acessados usando as APIs [GetTraceSummaries](#) e [BatchGetTraces](#), respectivamente. Além de segmentos e subsegmentos que você envia ao X-Ray, o serviço usa informações em subsegmentos para gerar segmentos inferidos e os adiciona ao rastreamento completo. Segmentos inferidos representam serviços e recursos subsequentes no mapa de rastreamento.

O X-Ray fornece um esquema JSON para documentos de segmentos. Você pode baixar o schema aqui: [xray-segmentdocument-schema-v1.0.0](#). Os campos e objetos listados no schema são descritos em mais detalhes nas seções a seguir.

Um subconjunto de campos de segmento é indexado pelo X-Ray para ser usado com expressões de filtro. Por exemplo, se você definir o campo `user` em um segmento para um identificador exclusivo, poderá pesquisar segmentos associados a usuários específicos no console do X-Ray ou usando a API `GetTraceSummaries`. Para obter mais informações, consulte [Usar expressões de filtro](#).

Quando você instrumenta o aplicativo com o X-Ray SDK, o SDK gera documentos de segmentos para você. Em vez de enviar documentos de segmentos diretamente ao X-Ray, o SDK transmite-os por meio de uma porta UDP local para o [daemon do X-Ray](#). Para obter mais informações, consulte [Enviar documentos de segmentos para o daemon do X-Ray](#).

Seções

- [Campos de segmento](#)
- [Subsegmentos](#)
- [Dados HTTP de solicitação](#)
- [Anotações](#)
- [Metadados](#)
- [Dados de recursos da AWS](#)
- [Erros e exceções](#)
- [Consultas SQL](#)

Campos de segmento

Um segmento registra informações de rastreamento sobre uma solicitação que o seu aplicativo atende. No mínimo, um segmento registra nome, ID, horário de início, ID de rastreamento e horário de término da solicitação.

Example Segmento completo mínimo

```
{
  "name" : "example.com",
  "id" : "70de5b6f19ff9a0a",
  "start_time" : 1.478293361271E9,
```

```
"trace_id" : "1-581cf771-a006649127e371903a2de979",  
"end_time" : 1.478293361449E9  
}
```

Os seguintes campos são obrigatórios ou condicionalmente necessários, por segmento.

Note

Os valores devem ser strings (até 250 caracteres), a menos que indicado o contrário.

Campos obrigatórios de segmento

- **name**: o nome lógico do serviço que processou a solicitação, com até 200 caracteres. Por exemplo, o nome do aplicativo ou o nome de domínio. Os nomes podem conter letras unicode, números e espaço em branco e os seguintes símbolos: `_`, `.`, `:`, `/`, `%`, `&`, `#`, `=`, `+`, `\`, `-`, `@`
- **id**: um identificador de 64 bits para o segmento, exclusivo entre segmentos no mesmo rastreamento, com 16 dígitos hexadecimais.
- **trace_id**: um identificador exclusivo que conecta todos os segmentos e subsegmentos provenientes de uma única solicitação do cliente.

Formato do ID de rastreamento do X-Ray

Um **trace_id** do X-Ray consiste em três números separados por hifens. Por exemplo, `1-58406520-a006649127e371903a2de979`. Isso inclui:

- O número da versão, que é 1.
- A hora da solicitação original, em horário epoch Unix, com 8 dígitos hexadecimais.

Por exemplo, 10h no dia 1º de dezembro de 2016 PST equivale a 1480615200 segundos em horário epoch ou a 58406520 em dígitos hexadecimais.

- Um identificador globalmente exclusivo de 96 bits para o rastreamento com 24 dígitos hexadecimais.

Note

O X-Ray agora aceita IDs de rastreamento criados usando o OpenTelemetry ou qualquer outro framework que esteja em conformidade com a [especificação W3C Trace Context](#). Um ID de rastreamento do W3C deve ser formatado conforme o ID de

rastreamento do X-Ray ao ser enviado ao X-Ray. Por exemplo, o ID de rastreamento 4efaaf4d1e8720b39541901950019ee5 do W3C deve ser formatado como 1-4efaaf4d-1e8720b39541901950019ee5 quando enviado ao X-Ray. Os IDs de rastreamento do X-Ray incluem o carimbo de data e hora da solicitação original no horário epoch Unix, mas isso não é necessário ao enviar IDs de rastreamento do W3C no formato do X-Ray.

Segurança de ID de Rastreamento

IDs de rastreamento são visíveis nos [cabeçalhos de resposta](#). Gere IDs de rastreamento com um algoritmo aleatório seguro para garantir que invasores não possam calcular futuras IDs de rastreamento e enviar solicitações com aqueles IDs para seu aplicativo.

- **start_time**: um número que representa o momento em que o segmento foi criado, em segundos de ponto flutuante na hora de época. Por exemplo, 1480615200.010 ou 1.480615200010E9. Use o máximo de casas decimais que precisar. A resolução em microssegundos é recomendada quando disponível.
- **end_time**: um número que representa o momento em que o segmento foi encerrado. Por exemplo, 1480615200.090 ou 1.480615200090E9. Especifique um **end_time** ou **in_progress**.
- **in_progress**: um valor booleano, definido como **true**, em vez de especificar um **end_time** para registrar que um segmento foi iniciado e não foi concluído. Envie um segmento em andamento quando seu aplicativo receber uma solicitação que levará muito tempo para atender, para rastrear o recebimento da solicitação. Quando a resposta é encaminhada, envie o segmento completo para substituir o segmento em andamento. Envie apenas um segmento completo e um ou nenhum segmento em andamento, por solicitação.

Nomes de serviço

O nome de um segmento deve corresponder ao nome de domínio ou nome lógico do serviço que gera o segmento. No entanto, isso não é aplicado. Qualquer aplicativo que tenha permissão para [PutTraceSegments](#) pode enviar segmentos com qualquer nome.

Os seguintes campos são opcionais para segmentos.

Campos opcionais de segmento

- `service`: um objeto com informações sobre o aplicativo.
 - `version`: uma string que identifica a versão do aplicativo que atendeu à solicitação.
- `user`: uma string que identifica o usuário que enviou a solicitação.
- `origin`: o tipo de recurso da AWS que executa o aplicativo.

Valores suportados

- `AWS::EC2::Instance`: uma instância do Amazon EC2.
- `AWS::ECS::Container`: um contêiner do Amazon ECS.
- `AWS::ElasticBeanstalk::Environment`: um ambiente do Elastic Beanstalk.

Quando vários valores são aplicáveis à seu aplicativo, use o que é mais específico. Por exemplo, um ambiente do Docker de vários contêineres no Elastic Beanstalk executa o aplicativo em um contêiner do Amazon ECS, que, por sua vez, é executado em uma instância do Amazon EC2. Neste caso, você define a origem para `AWS::ElasticBeanstalk::Environment` como o ambiente pai dos outros dois recursos.

- `parent_id`: um ID de subsegmento que você especifica se a solicitação tiver se originado de um aplicativo instrumentada. O X-Ray SDK adiciona a ID do subsegmento principal ao [cabeçalho de rastreamento](#) para chamadas HTTP subsequentes. No caso de subsegmentos aninhados, um subsegmento pode ter um segmento ou um subsegmento como seu pai.
- `http`: objetos [http](#) com informações sobre a solicitação HTTP original.
- `aws`: objeto [aws](#) com informações sobre o recurso da AWS no qual o aplicativo atendeu à solicitação.
- `error`, `throttle`, `fault` e `cause`: campos de [erro](#) que indicam um erro ocorrido e que incluem informações sobre a exceção que causou o erro.
- `annotations` – [annotations](#) o objeto com os pares de chave-valor que você deseja que o X-Ray indexe para pesquisa.
- `metadata` – [metadata](#) objeto com qualquer dado adicional que você deseja armazenar no segmento.
- `subsegments`: matriz de objetos [subsegment](#).

Subsegmentos

Você pode criar subsegmentos para registrar chamadas a recursos e Serviços da AWS que você faz com o SDK da AWS, chamadas de APIs da web HTTP internas ou externas ou consultas SQL de banco de dados. Você também pode criar subsegmentos para depurar ou anotar blocos de código em seu aplicativo. Subsegmentos podem conter outros subsegmentos, portanto, um subsegmento personalizado que registra metadados sobre uma chamada de função interna pode conter outros subsegmentos personalizados e subsegmentos para chamadas de downstream.

Um subsegmento registra uma chamada subsequente do ponto de vista do serviço que faz a chamada. O X-Ray usa subsegmentos para identificar serviços subsequentes que não enviam segmentos e criam entradas para eles no gráfico de serviço.

Um subsegmento pode ser incorporado em um documento de segmento completo ou enviado de forma independente. Envie subsegmentos separadamente para chamadas downstream de rastreamento de forma assíncrona para solicitações de longo prazo, ou para evitar exceder o tamanho máximo do documento de segmento.

Example Segmento com subsegmento incorporado

Um subsegmento independente tem um `type` de `subsegment` e uma `parent_id` que identifica o segmento pai.

```
{
  "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
      }
    }
  }
]
}

```

Para solicitações de longa duração, você pode enviar um segmento em andamento para notificar o X-Ray de que a solicitação foi recebida e, em seguida, enviar subsegmentos separadamente para rastreá-los antes de concluir a solicitação original.

Example Segmento em andamento

```

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

```

Example Subsegmento independente

Um subsegmento independente tem um `type` de `subsegment`, um `trace_id` e um `parent_id` que identifica o segmento pai.

```

{

```



```
"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
  }
}
```

Quando a solicitação for concluída, feche o segmento reenviando-o com um `end_time`. O segmento completo substituirá o segmento em andamento.

Você também pode enviar subsegmentos separadamente para solicitações concluídas que acionaram fluxos de trabalho assíncronos. Por exemplo, uma API da Web pode retornar uma resposta OK 200 imediatamente antes de iniciar o trabalho que o usuário solicitou. Você pode enviar um segmento completo para o X-Ray assim que a resposta for enviada e, depois, subsegmentos para o trabalho concluído em um momento posterior. Assim como ocorre com segmentos, você também pode enviar um fragmento de subsegmento para registrar que o subsegmento foi iniciado e, em seguida, sobregravá-lo com um subsegmento completo assim que a chamada de downstream for concluída.

Os seguintes campos são obrigatórios ou condicionalmente necessários, por subsegmento.

Note

Os valores são strings de até 250 caracteres, a menos que indicado o contrário.

Campos obrigatórios de subsegmento

- **id**: um identificador de 64 bits para o subsegmento, exclusivo entre segmentos no mesmo rastreamento, com 16 dígitos hexadecimais.
- **name**: o nome lógico do subsegmento. Para chamadas de downstream, nomeie o subsegmento após o recurso ou serviço chamado. Para subsegmentos personalizados, nomeie o subsegmento depois do código que ele instrumenta (por exemplo, um nome de função).
- **start_time**: um número que representa o momento em que o subsegmento foi criado, em segundos de ponto flutuante no horário de época, com precisão de milissegundos. Por exemplo, 1480615200.010 ou 1.480615200010E9.
- **end_time**: um número que representa o momento em que o subsegmento foi encerrado. Por exemplo, 1480615200.090 ou 1.480615200090E9. Especifique um **end_time** ou **in_progress**.
- **in_progress**: um valor booleano, definido como **true**, em vez de especificar um **end_time** para registrar que um subsegmento foi iniciado e não foi concluído. Envie apenas um subsegmento completo e um ou nenhum subsegmento em andamento, por solicitação de downstream.
- **trace_id**: ID de rastreamento do segmento principal do subsegmento. Obrigatório apenas ao enviar um subsegmento separadamente.

Formato do ID de rastreamento do X-Ray

Um **trace_id** do X-Ray consiste em três números separados por hifens. Por exemplo, 1-58406520-a006649127e371903a2de979. Isso inclui:

- O número da versão, que é 1.
- A hora da solicitação original, em horário epoch Unix, com 8 dígitos hexadecimais.

Por exemplo, 10h no dia 1º de dezembro de 2016 PST equivale a 1480615200 segundos em horário epoch ou a 58406520 em dígitos hexadecimais.

- Um identificador globalmente exclusivo de 96 bits para o rastreamento com 24 dígitos hexadecimais.

Note

O X-Ray agora aceita IDs de rastreamento criados usando o OpenTelemetry ou qualquer outro framework que esteja em conformidade com a [especificação W3C Trace Context](#). Um ID de rastreamento do W3C deve ser formatado conforme o ID de

rastreamento do X-Ray ao ser enviado ao X-Ray. Por exemplo, o ID de rastreamento 4efaaf4d1e8720b39541901950019ee5 do W3C deve ser formatado como 1-4efaaf4d-1e8720b39541901950019ee5 quando enviado ao X-Ray. Os IDs de rastreamento do X-Ray incluem o carimbo de data e hora da solicitação original no horário epoch Unix, mas isso não é necessário ao enviar IDs de rastreamento do W3C no formato do X-Ray.

- `parent_id`: ID do segmento principal do subsegmento. Obrigatório apenas ao enviar um subsegmento separadamente. No caso de subsegmentos aninhados, um subsegmento pode ter um segmento ou um subsegmento como seu pai.
- `type`: `subsegment`. Obrigatório apenas ao enviar um subsegmento separadamente.

Os seguintes campos são opcionais para subsegmentos.

Campos opcionais de subsegmento

- `namespace`: `aws` para chamadas de SDK da AWS; `remote` para outras chamadas subsequentes.
- `http`: objeto [http](#) com informações sobre uma chamada HTTP de saída.
- `aws`: objeto [aws](#) com informações sobre o recurso da AWS subsequente que o aplicativo chamou.
- `error`, `throttle`, `fault` e `cause`: campos de [erro](#) que indicam um erro ocorrido e que incluem informações sobre a exceção que causou o erro.
- `annotations`: o objeto [annotations](#) com os pares de chave-valor que você deseja que o X-Ray indexe para pesquisa.
- `metadata`: objeto [metadata](#) com qualquer dado adicional que você deseja armazenar no segmento.
- `subsegments`: matriz de objetos [subsegment](#).
- `precursor_ids`: matriz de IDs de subsegmentos que identifica subsegmentos com o mesmo pai que foram concluídos antes deste subsegmento.

Dados HTTP de solicitação

Use um bloco HTTP para registrar detalhes sobre uma solicitação HTTP que seu aplicativo atendeu (em um segmento) ou que seu aplicativo realizou para uma API HTTP downstream (em um subsegmento). A maioria dos campos deste objeto são mapeados para informações encontrados em uma solicitação e uma resposta HTTP.

http

Todos os campos são opcionais.

- **request:** informações sobre uma solicitação.
 - **method:** o método de solicitação. Por exemplo, GET.
 - **url:** o URL completo da solicitação, compilado com base no protocolo, no nome do host e no caminho da solicitação.
 - **user_agent:** a string do agente de usuário do cliente do solicitante.
 - **client_ip:** o endereço IP do solicitante. Pode ser recuperado do `Source Address` do pacote IP ou, para solicitações encaminhadas, a partir de um `X-Forwarded-For` cabeçalho.
 - **x_forwarded_for:** (apenas para segmentos) um booliano indicando que o `client_ip` foi lido de um cabeçalho `X-Forwarded-For` e não é confiável, pois ele pode ter sido falsificado.
 - **traced:** (apenas para subsegmentos) um booliano indicando que a chamada subsequente é para outro serviço rastreado. Se este campo estiver definido como `true`, o X-Ray considerará o rastreamento como dividido até que o serviço subsequente carregue um segmento com um `id` que corresponda ao `parent_id` do subsegmento que contém esse bloco.
- **response:** informações sobre uma resposta.
 - **status:** número inteiro indicando o status HTTP da resposta.
 - **content_length:** número inteiro indicando o tamanho do corpo da resposta em bytes.

Ao instrumentar uma chamada subsequente de API da web, registre um subsegmento com informações sobre a solicitação e a resposta HTTP. O X-Ray usa o subsegmento para gerar um segmento inferido para a API remota.

Example Segmento para chamada HTTP servido por um aplicativo em execução no Amazon EC2

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

```

    "instance_id": "i-0b5a4678fc325bg98"
  },
  "xray": {
    "sdk_version": "2.11.0 for Java"
  },
},
"http": {
  "request": {
    "method": "POST",
    "client_ip": "78.255.233.48",
    "url": "http://www.example.com/api/user",
    "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101
Firefox/45.0",
    "x_forwarded_for": true
  },
  "response": {
    "status": 200
  }
}

```

Example Subsegmento para uma chamada HTTP downstream

```

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

```

Example Segmento inferido para uma chamada HTTP de downstream

```

{

```

```
{
  "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
}
```

Anotações

Os segmentos e subsegmentos podem incluir um objeto `annotations` contendo um ou mais campos que o X-Ray indexa para serem usados com expressões de filtro. Os campos podem ter string, número ou valores boolianos (sem objetos ou matrizes). O X-Ray indexa até cinquenta anotações por rastreamento.

Example Segmento para chamada HTTP com anotações

```
{
  "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
  }
}
}

```

As chaves devem ser alfanuméricas para funcionar com filtros. Sublinhado é permitido. Outros símbolos e espaço em branco não são permitidos.

Metadados

Os segmentos e subsegmentos podem incluir um objeto metadado contendo um ou mais campos com valores de qualquer tipo, incluindo objetos e matrizes. O X-Ray não indexa metadados, e os valores podem ser de qualquer tamanho, desde que o documento de segmentos não ultrapasse o tamanho máximo (64 kB). Você pode visualizar os metadados no documento de segmento completo retornado pela API [BatchGetTraces](#). As chaves de campo (debug no exemplo a seguir) que começam com `AWS.` são reservadas para serem usadas por SDKs e clientes fornecidos pela AWS.

Exemplo Subsegmento personalizado com metadados

```

{
  "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": "3AIENM5J4ELQ3SP0DHKBIRVIC3VV4KQNS05AEMVJF66Q9ASUAAJG",
      "resource_names": [
        "scorekeep-user"
      ]
    }
  }
]
}

```

Dados de recursos da AWS

Para segmentos, o objeto `aws` contém informações sobre o recurso no qual seu aplicativo é executado. Vários campos podem ser aplicados a um único recurso. Por exemplo, um aplicativo em execução em um ambiente do Docker de vários contêineres no Elastic Beanstalk poderia ter informações sobre a instância do Amazon EC2, o contêiner do Amazon ECS em execução na instância e o próprio ambiente do Elastic Beanstalk.

aws (Segmentos)

Todos os campos são opcionais.

- `account_id`: se o aplicativo envia segmentos para uma outra Conta da AWS, registre o ID da conta que está executando o aplicativo.

- `cloudwatch_logs`: conjunto de objetos que descrevem um único grupo de logs do CloudWatch.
 - `log_group`: o nome do grupo de logs do CloudWatch.
 - `arn`: o ARN do grupo de logs do CloudWatch.
- `ec2`: informações sobre uma instância do EC2.
 - `instance_id`: o ID da instância do EC2.
 - `instance_size`: o tipo da instância do EC2.
 - `ami_id`: o ID da imagem de máquina da Amazon.
 - `availability_zone`: a zona de disponibilidade na qual a instância está sendo executada.
- `ecs`: informações sobre um contêiner do Amazon ECS.
 - `container`: o nome do host do contêiner.
 - `container_id`: o ID completo do contêiner.
 - `container_arn`: o ARN da instância de contêiner.
- `eks`: informações sobre um cluster do Amazon EKS.
 - `pod`: o nome do host do pod do EKS.
 - `cluster_name`: o nome do cluster do EKS.
 - `container_id`: o ID completo do contêiner.
- `elastic_beanstalk`: informações sobre um ambiente do Elastic Beanstalk. Você encontra essas informações em um arquivo chamado `/var/elasticbeanstalk/xray/environment.conf` nas plataformas mais recentes do Elastic Beanstalk.
 - `environment_name`: o nome do ambiente.
 - `version_label`: o nome da versão do aplicativo que está implantada no momento na instância que atendeu à solicitação.
 - `deployment_id`: um número que indica o ID da última implantação bem-sucedida na instância que atendeu à solicitação.
- `xray`: metadados sobre o tipo e a versão da instrumentação usada.
 - `auto_instrumentation`: booliano que indica se a instrumentação automática foi usada (por exemplo, o agente do Java).
 - `sdk_version`: a versão do SDK ou do agente que está sendo usada.
 - `sdk`: o tipo de SDK.

Example Bloco da AWS com plug-ins

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

Para subsegmentos, registre informações sobre os recursos e Serviços da AWS que o aplicativo acessa. O X-Ray usa essas informações para criar segmentos inferidos que representam os serviços subsequentes no mapa de serviço.

aws (Subsegmentos)

Todos os campos são opcionais.

- **operation**: o nome da ação de API invocada para um recurso ou AWS service (Serviço da AWS).
- **account_id**: se o aplicativo acessa recursos em uma outra conta ou envia segmentos para uma conta diferente, registre o ID da conta que controla o recurso da AWS que o aplicativo acessou.
- **region**: se o recurso estiver em uma região diferente do aplicativo, registre a região. Por exemplo, `us-west-2`.
- **request_id**: identificador exclusivo da solicitação.

- `queue_url`: para operações em uma fila do Amazon SQS, o URL da fila.
- `table_name`: para operações em uma tabela do DynamoDB, o nome da tabela.

Example Subsegmento para uma chamada ao DynamoDB para salvar um item

```
{
  "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": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Erros e exceções

Quando ocorrer um erro, você pode registrar detalhes sobre o erro e as exceções que foram gerados. Registre erros em segmentos quando o seu aplicativo retornar um erro para o usuário e em subsegmentos quando uma chamada de downstream retornar um erro.

tipos de erro

Defina um ou mais dos seguintes campos do `true` para indicar que ocorreu um erro. Vários tipos podem ser aplicados se erros ocorrerem. Por exemplo, um 429 Too Many Requests erro de uma chamada de downstream pode fazer com que seu aplicativo retorne 500 Internal Server Error, caso em que os três tipos são aplicáveis.

- `error`: um booleano indicando a ocorrência de um erro de cliente (o código de status de resposta foi 4XX Erro de cliente).

- **throttle**: um booliano indicando que uma solicitação foi suspensa (o código de status de resposta foi 429 Solicitações em excesso).
- **fault**: um booliano indicando a ocorrência de um erro de servidor (o código de status de resposta foi 5XX Erro de servidor).

Indique a causa do erro, incluindo o objeto da causa no segmento ou subsegmento.

cause

Uma causa pode ser um ID de exceção de 16 caracteres ou um objeto com os seguintes campos:

- **working_directory**: o caminho completo do diretório de trabalho quando a exceção ocorreu.
- **paths**: a matriz de caminhos para bibliotecas ou módulos em uso quando a exceção ocorreu.
- **exceptions**: a matriz de objetos de exceção.

Incluir informações detalhadas sobre o erro em um ou mais objetos de exceção.

exception

Todos os campos são opcionais.

- **id**: um identificador de 64 bits para a exceção, exclusivo entre segmentos no mesmo rastreamento, com 16 dígitos hexadecimais.
- **message**: a mensagem de exceção.
- **type**: o tipo de exceção.
- **remote**: um booliano indicando que a exceção foi causada por um erro retornado por um serviço subsequente.
- **truncated**: um inteiro indicando o número de estruturas de pilhas que são omitidas da `stack`.
- **skipped**: um inteiro indicando o número de exceções que foram ignoradas entre essa exceção e a exceção secundária, ou seja, a exceção que ela causou.
- **cause**: o ID da exceção principal, ou seja, a exceção que causou essa exceção.
- **stack**: a matriz de objetos `stackFrame`.

Se disponíveis, registre informações sobre a pilha de chamadas nos objetos `stackFrame`.

stackFrame

Todos os campos são opcionais.

- `path`: o caminho relativo para o arquivo.
- `line`: a linha no arquivo.
- `label`: a função ou o nome do método.

Consultas SQL

Você pode criar subsegmentos para consultas que seu aplicativo faz em um banco de dados SQL.

sql

Todos os campos são opcionais.

- `connection_string`: para o SQL Server ou outras conexões de banco de dados que não usam strings de conexão de URL, registre a string de conexão sem as senhas.
- `url`: para uma conexão de banco de dados que usa uma string de conexão, registre o URL sem as senhas.
- `sanitized_query`: a consulta de banco de dados, com os valores fornecidos pelo usuário removidos ou substituídos por um espaço reservado.
- `database_type`: o nome do mecanismo de banco de dados.
- `database_version`: o número da versão do mecanismo de banco de dados.
- `driver_version`: o nome e o número da versão do driver do mecanismo de banco de dados que o aplicativo utiliza.
- `user`: o nome de usuário do banco de dados.
- `preparation`: `call` se a consulta usou uma `PreparedCall`; `statement` se a consulta usou uma `PreparedStatement`.

Example Subsegmento com uma consulta SQL

```
{
  "id": "3fd8634e78ca9560",
  "start_time": 1484872218.696,
  "end_time": 1484872218.697,
  "name": "ebdb@aawijb5u25wdoy.cpamxznpdoq8.us-west-2.rds.amazonaws.com",
```

```
"namespace": "remote",
"sql" : {
  "url": "jdbc:postgresql://aawijb5u25wdoy.cpamxznpdq8.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

Conceitos do

O AWS X-Ray recebe dados de serviços como segmentos. Em seguida, o X-Ray agrupa segmentos que tenham uma solicitação em comum em rastreamentos. O X-Ray processa os rastreamentos para gerar um gráfico de serviço que apresenta uma representação visual da aplicação.

Conceitos

- [Segmentos](#)
- [Subsegmentos](#)
- [Gráfico de serviço](#)
- [Rastreamentos](#)
- [Amostragem](#)
- [Cabeçalho de rastreamento](#)
- [Expressões de filtro](#)
- [Grupos](#)
- [Anotações e metadados](#)
- [Erros, falhas e exceções](#)

Segmentos

Os recursos de computação que executam a lógica do aplicativo enviam dados sobre o trabalho como segmentos. Um segmento fornece o nome do recurso, os detalhes sobre a solicitação e os detalhes sobre o trabalho realizado. Por exemplo, quando alcança o aplicativo, uma solicitação HTTP pode registrar os seguintes dados sobre:

- O host: nome do host, alias ou endereço IP
- A solicitação: método, endereço do cliente, caminho e agente do usuário
- A resposta: status, conteúdo
- O trabalho realizado: hora de início e de término, subsegmentos
- Problemas que ocorrem: [erros, falhas e exceções](#), incluindo captura automática das pilhas de exceções.

Segment details: Scorekeep



Overview	Resources	Annotations	Metadata	Exceptions	SQL
Overview Subsegment ID 1-12345678-5120cbe96265dfa965cba1ac-556f7a611a12900FF Name Scorekeep Origin AWS::ECS::Container			Time Start Time 2023-06-23 20:34:58.099 (UTC) End Time 2023-06-23 20:34:58.110 (UTC) Duration 11ms	Errors and faults Error false Fault false	Requests & Response Request url http://scorekeep.us-west-2.elb.amazonaws.com/api/game/ Request method GET Response code 200

O X-Ray SDK coleta informações dos cabeçalhos de solicitação e de resposta, o código na aplicação e metadados sobre os recursos da AWS em que ele é executado. Você escolhe os dados a serem coletados modificando a configuração ou o código da aplicação para instrumentar solicitações recebidas, solicitações subsequentes e clientes do SDK da AWS.

Solicitações encaminhadas

Se um balanceador de carga ou outro intermediário encaminhar uma solicitação para a aplicação, o X-Ray obterá o IP do cliente do cabeçalho X-Forwarded-For na solicitação em vez do IP de origem no pacote IP. O IP do cliente registrado para uma solicitação encaminhada pode ser falsificado; portanto, não é digno de confiança.

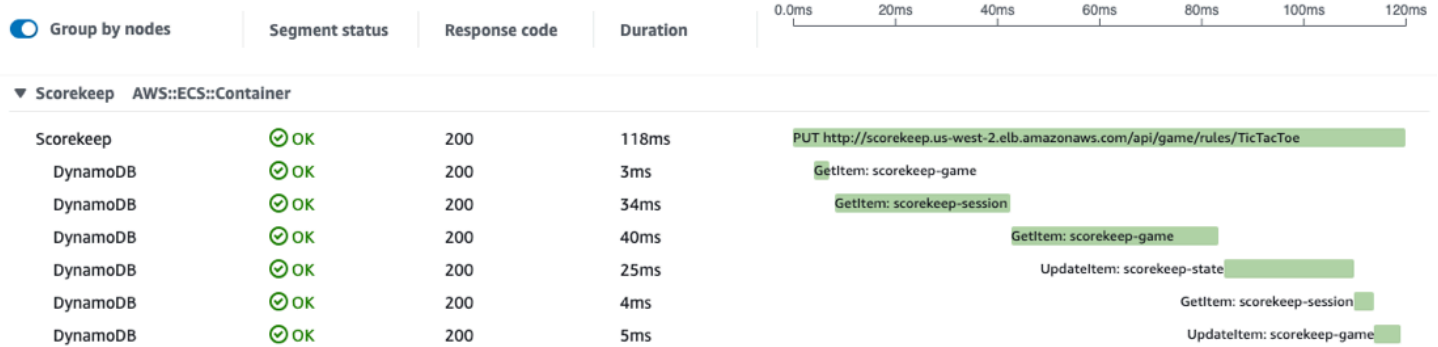
Você pode usar o X-Ray SDK para registrar informações adicionais, como [anotações e metadados](#). Para obter mais detalhes sobre a estrutura e informações registradas em segmentos e subsegmentos, consulte [AWS X-Ray Documentos de segmento do](#). Os documentos de segmentos podem ter até 64 kB de tamanho.

Subsegmentos

Um segmento pode dividir os dados sobre o trabalho feito em subsegmentos. Os subsegmentos fornecem informações de tempo mais granulares e detalhes sobre chamadas de downstream feitas pelo aplicativo para atender à solicitação original. Um subsegmento pode conter detalhes adicionais sobre uma chamada para um AWS service (Serviço da AWS), uma API HTTP externa ou um banco

de dados SQL. Você pode até mesmo definir subsegmentos arbitrários para instrumentar funções ou linhas de código específicas em seu aplicativo.

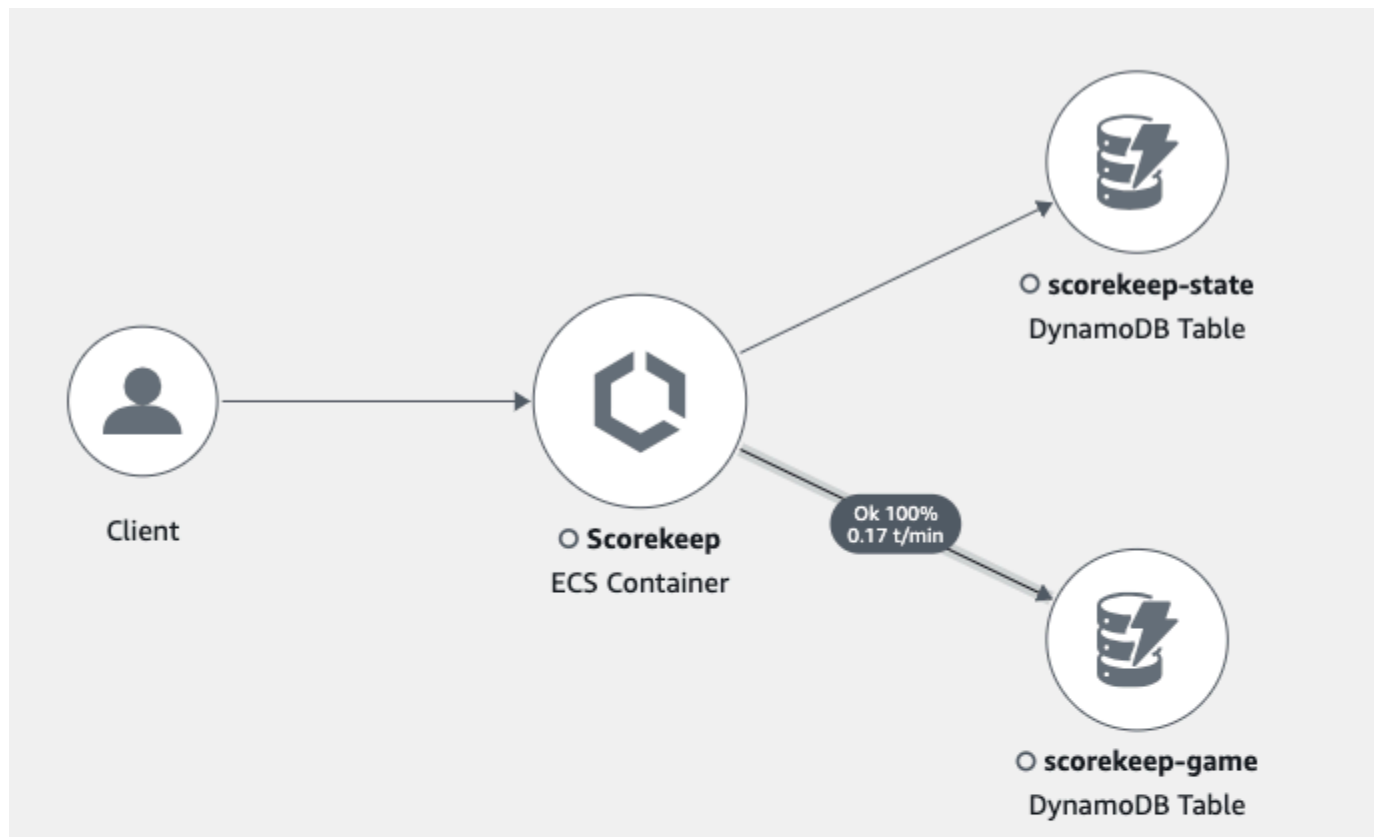
Segments Timeline [Info](#)



Para serviços que não enviam seus próprios segmentos, como o Amazon DynamoDB, o X-Ray usa subsegmentos para gerar segmentos inferidos e nós subsequentes no mapa de rastreamento. Isso permite que você veja todas as suas dependências downstream, mesmo que elas não ofereçam suporte a rastreamento ou sejam externas.

Os subsegmentos representam a visualização de seu aplicativo de uma chamada downstream como um cliente. Se o serviço downstream também for instrumentado, o segmento que ele envia substitui o segmento deduzido gerado no subsegmento do cliente upstream. O nó no gráfico do serviço sempre usa informações do segmento do serviço, se disponíveis, enquanto a borda entre os dois nós usa o subsegmento do serviço upstream.

Por exemplo, quando você chama o DynamoDB com um cliente instrumentado do SDK da AWS, o X-Ray SDK registra um subsegmento para essa chamada. O DynamoDB não envia um segmento, de modo que o segmento inferido no rastreamento, o nó do DynamoDB no gráfico de serviço e a borda entre o serviço e o DynamoDB contêm informações do subsegmento.

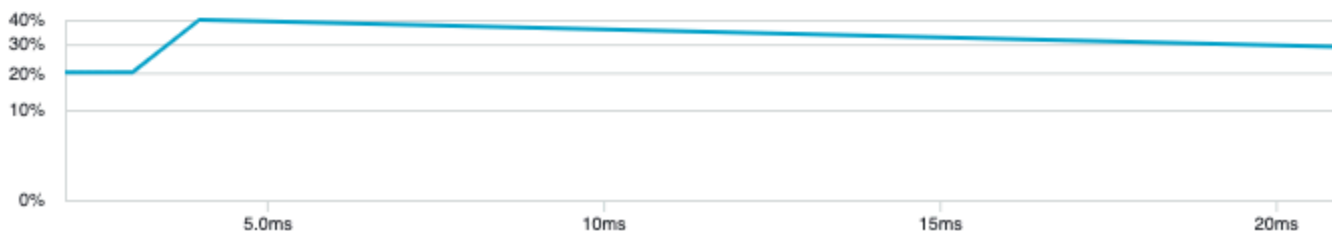


▼ Edge details

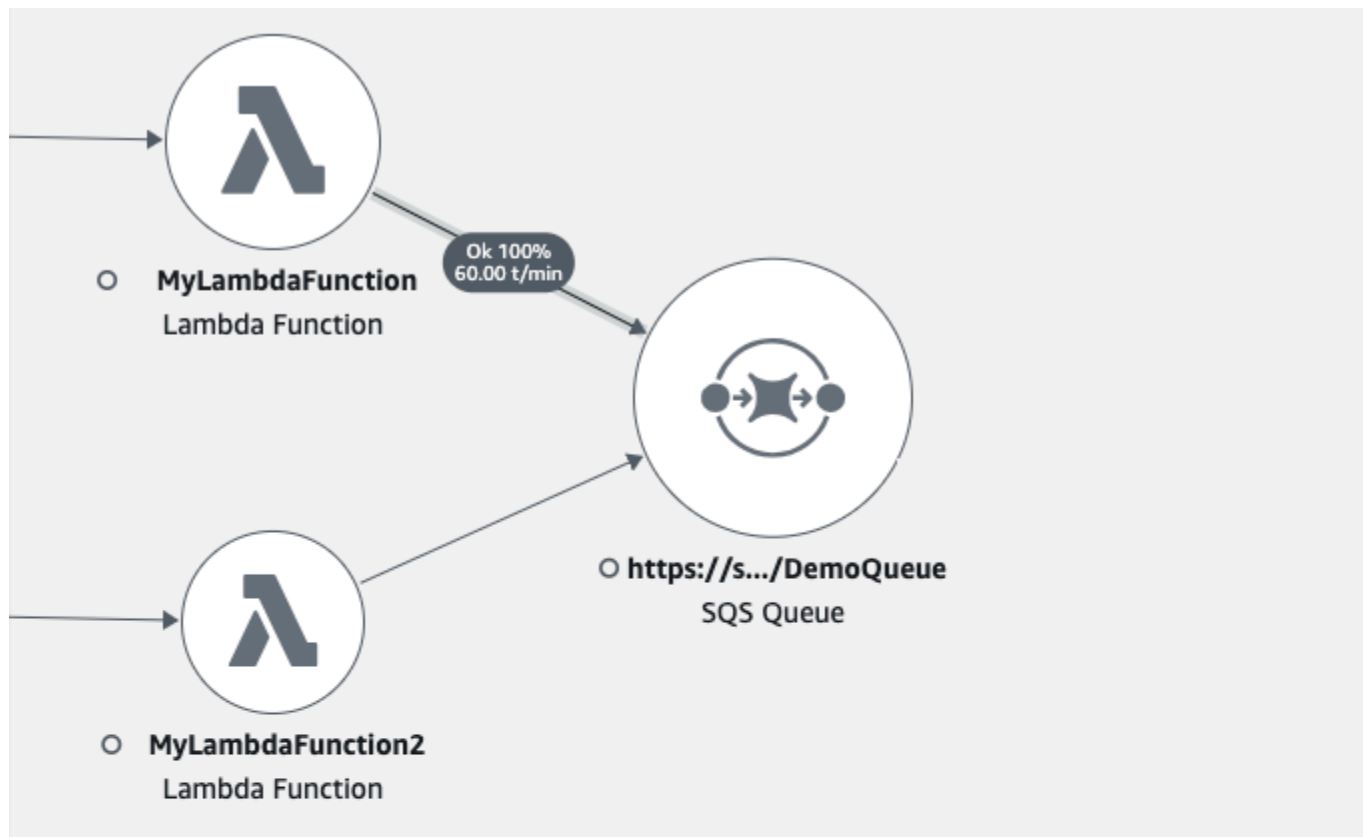
Source: Scorekeep Destination: scorekeep-game

Response time distribution filter

To filter traces by response time, select the corresponding area of the chart.



Quando você chama outro serviço instrumentado com um aplicativo instrumentado, o serviço downstream envia seu próprio segmento para gravar sua visualizado da mesma chamada que o serviço upstream gravou em um segmento. No gráfico do serviço, os dois nós de serviço contêm o tempo e as informações de erro dos segmentos desses serviços, enquanto a borda entre eles contém informações do subsegmento do serviço upstream.



▼ Edge details

Source: MyLambdaFunction Destination: <https://sqs.us-west-2.amazonaws.com/MySQSQueue>

Response time distribution filter

To filter traces by response time, select the corresponding area of the chart.



Os dois pontos de vista são úteis, uma vez que o serviço downstream grava precisamente quando iniciou e terminou o trabalho na solicitação, e o serviço upstream grava a latência da viagem de ida e volta, incluindo o tempo que a solicitação gastou viajando entre os dois serviços.

Gráfico de serviço

O X-Ray usa os dados que a aplicação envia para gerar um gráfico de serviço. Cada recurso da AWS que envia dados ao X-Ray é exibido como um serviço no gráfico. Margens conectam os serviços que funcionam juntos para atender a solicitações. Margens conectam clientes ao aplicativo e o aplicativo a serviços e recursos de downstream usados por ele.

Nomes de serviço

O nome de um segmento deve corresponder ao nome de domínio ou nome lógico do serviço que gera o segmento. No entanto, isso não é aplicado. Qualquer aplicação que tenha permissão para [PutTraceSegments](#) pode enviar segmentos com qualquer nome.

Gráfico de serviço é um documento JSON que contém informações sobre os serviços e os recursos que compõem o aplicativo. O console do X-Ray usa o gráfico de serviço para gerar uma visualização ou um mapa de serviço.



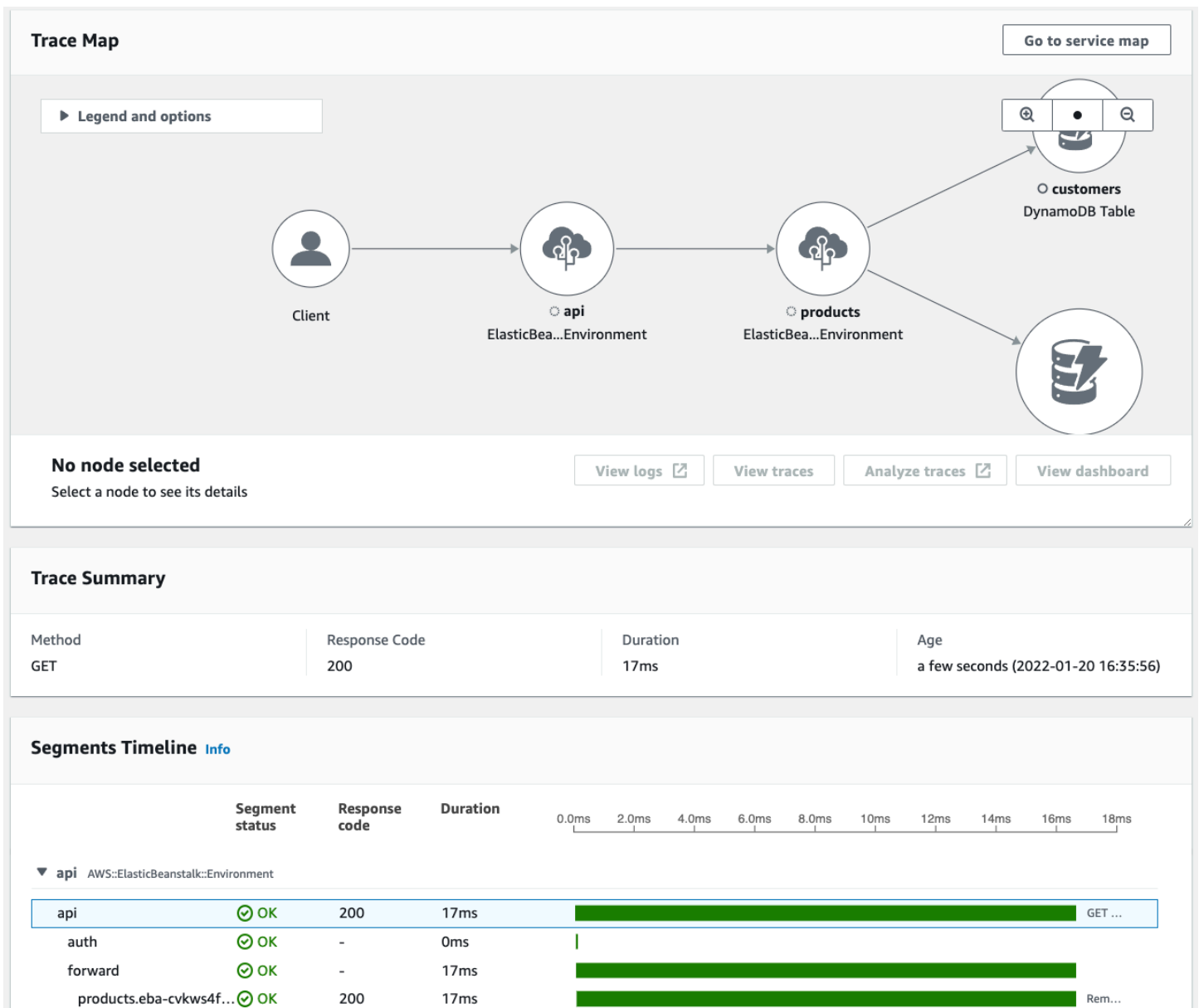
Para uma aplicação distribuída, o X-Ray combina nós de todos os serviços que processam solicitações com a mesma ID de rastreamento em um único gráfico de serviço. O primeiro serviço adicionado pela solicitação inclui um [cabeçalho de rastreamento](#) propagado entre o front-end e os serviços chamados.

Por exemplo, o [Scorekeep](#) executa uma API da web que chama um microserviço (uma função do AWS Lambda) para gerar um nome aleatório usando uma biblioteca Node.js. O X-Ray SDK para Java gera um ID de rastreamento e o inclui em chamadas para o Lambda. O Lambda envia dados de rastreamento e passa o ID de rastreamento para a função. O X-Ray SDK para Node.js também usa o ID de rastreamento para enviar dados. Dessa forma, os nós para a API, o serviço Lambda e a função do Lambda são todos exibidos como nós separados, embora conectados, no mapa de rastreamento.

O gráfico de serviço é retido por trinta dias.

Rastreamentos

Uma ID de rastreamento acompanha o caminho de uma solicitação pelo aplicativo. Um rastreamento coleta todos os segmentos gerados por uma única solicitação. Essa solicitação costuma ser uma solicitação HTTP GET ou POST que percorre um balanceador de carga, chega ao código da aplicação e gera chamadas subsequentes para outros serviços da AWS ou APIs da web externas. O primeiro serviço compatível com o qual a solicitação HTTP interage adiciona um cabeçalho de ID de rastreamento à solicitação e o propaga downstream para rastrear a latência, a disposição e outros dados de solicitação.



Consulte os [Preços do AWS X-Ray](#) para obter informações sobre como os rastreamentos do X-Ray são cobrados. Os dados de rastreamento são retidos por trinta dias.

Amostragem

Para garantir um rastreamento eficiente e fornecer uma amostra representativa das solicitações atendidas pela aplicação, o X-Ray SDK aplica um algoritmo de amostragem para determinar quais solicitações são rastreadas. O padrão é registrar a primeira solicitação a cada segundo e 5% de todas as solicitações adicionais.

Para evitar cobranças incorridas para o serviço quando você está começando a usá-lo, a taxa de amostragem padrão é conservadora. Você pode configurar o X-Ray para modificar a regra de amostragem padrão e configurar regras adicionais que aplicam a amostragem com base nas propriedades do serviço ou da solicitação.

Por exemplo, pode ser conveniente desabilitar a amostragem e rastrear todas as solicitações para chamadas que modificam um estado ou lidam com usuários ou transações. Para chamadas de somente leitura de alto volume, como a sondagem de plano de fundo, as verificações de integridade ou a manutenção da conexão, você pode coletar amostras a uma taxa baixa e ainda obter dados suficientes para ver qualquer problema que surja.

Para obter mais informações, consulte [Configurar regras de amostragem](#).

Cabeçalho de rastreamento

Todas as solicitações são rastreadas até um mínimo configurável. Após atingir esse mínimo, uma porcentagem de solicitações será rastreada para evitar custos desnecessários. A ID de rastreamento e a decisão de amostragem são adicionadas a solicitações HTTP em cabeçalhos de rastreamento chamados `X-Amzn-Trace-Id`. O primeiro serviço integrado do X-Ray atingido pela solicitação adiciona um cabeçalho de rastreamento, que é lido pelo X-Ray SDK e incluído na resposta.

Exemplo Cabeçalho de rastreamento com ID de rastreamento raiz e decisão de amostragem

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

Segurança do cabeçalho de rastreamento

Um cabeçalho de rastreamento pode se originar do X-Ray SDK, de um AWS service (Serviço da AWS) ou da solicitação do cliente. O aplicativo pode remover `X-Amzn-Trace-Id` de solicitações de entrada para evitar problemas causados por usuários adicionando IDs de rastreamento ou decisões de amostragem às solicitações.

O cabeçalho de rastreamento também pode conter um ID de segmento pai se a solicitação tiver se originado de um aplicativo instrumentado. Por exemplo, se sua aplicação chamar uma API da web HTTP subsequente com um cliente HTTP instrumentado, o X-Ray SDK adicionará o ID do segmento

da solicitação original ao cabeçalho de rastreamento da solicitação subsequente. Um aplicativo instrumentado que atende a solicitação de downstream pode registrar o ID de segmento pai para conectar as duas solicitações.

Example Cabeçalho de rastreamento com ID de rastreamento de raiz, ID de segmento pai e decisão de amostragem

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

Lineage pode ser anexado ao cabeçalho de rastreamento pelo Lambda e por outros Serviços da AWS como parte dos respectivos mecanismos de processamento e não deve ser usado diretamente.

Example Rastrear cabeçalho com Lineage

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

Expressões de filtro

Mesmo com amostragem, um aplicativo complexo gera uma grande quantidade de dados. O console do AWS X-Ray fornece uma visualização de fácil navegação do gráfico de serviço. Ele mostra informações de integridade e desempenho que ajudam a identificar problemas e oportunidades de otimização no seu aplicativo. Para o rastreamento avançado, você pode analisar rastreamentos de solicitações individuais, ou usar expressões de filtro para encontrar rastreamentos relacionados a caminhos ou usuários específicos.

Traces [Info](#)

5m 15m 30m

Find traces by typing a trace ID or query, build a query using the Query refiners section, or [choose a sample query](#). You can also type a trace ID here.

Run query 🟢 5 traces retrieved

Query refiners

Traces (5)
This table shows the most recent traces with an average response time of 0.16s. It shows as many as 1000 traces.

ID	Trace status	Timestamp	Response code	Response Time	Duration	HTTP Method
...561513004630e58c75c992ed	🟢 OK	3.4min (2023-08-16 17:39:20)	200	0.104s	0.104s	POST
...2e83714b7daac593167d2e73	🟢 OK	3.4min (2023-08-16 17:39:19)	200	0.07s	0.07s	POST
...54740787431329383155f154	🟢 OK	3.4min (2023-08-16 17:39:18)	200	0.1s	0.1s	POST

Grupos

Com a extensão de expressões de filtro, o X-Ray também comporta o recurso de grupo. Ao usar uma expressão de filtro, você pode definir critérios para aceitar rastreamentos no grupo.

Você pode chamar o grupo pelo nome ou pelo nome do recurso da Amazon (ARN) para gerar seu próprio gráfico de serviço, resumos de rastreamento e métricas do Amazon CloudWatch. Assim que um grupo é criado, os rastreamentos recebidos são verificados em relação à expressão de filtro do grupo, pois são armazenados no serviço X-Ray. As métricas referentes à quantidade de rastreamentos correspondentes a cada critério são publicadas no CloudWatch a cada minuto.

Atualizar a expressão de filtro de um grupo não altera os dados que já estão registrados. A atualização se aplica somente aos rastreamentos subsequentes. Isso pode resultar em um gráfico mesclado das expressões novas e antigas. Para evitar isso, exclua o grupo atual e crie um novo.

Note

Os grupos são faturados pelo número de rastreamentos recuperados que correspondem à expressão de filtro. Para obter mais informações, consulte [Preços do AWS X-Ray](#).

Para obter mais informações sobre grupos, consulte [Configurar grupos](#).

Anotações e metadados

Quando você instrumenta a aplicação, o X-Ray SDK registra informações sobre solicitações recebidas e enviadas, os recursos utilizados da AWS e a aplicação propriamente dita. É possível adicionar outras informações ao segmento documentado como anotações e metadados. Anotações e metadados são agregados no nível de rastreamento e podem ser adicionados a qualquer segmento ou subsegmento.

Anotações são pares de chave-valor simples indexados para serem usados com [expressões de filtro](#). Use anotações para registrar dados que você deseja usar para agrupar rastreamentos no console ou ao chamar a API [GetTraceSummaries](#).

O X-Ray indexa até cinquenta anotações por rastreamento.

Metadados são pares de chave-valor com valores de qualquer tipo, inclusive objetos e listas, mas que não são indexados. Use metadados para registrar dados que você deseja armazenar no rastreamento, mas não precisa usar para pesquisar rastreamentos.

Você pode visualizar anotações e metadados na janela de detalhes do segmento ou subsegmento, na página [Detalhes de rastreamento](#) no console do CloudWatch.

▼ DynamoDB AWS::DynamoDB::Table					
DynamoDB	✓ OK	200	9ms	GetItem: scorekeep-session	
DynamoDB	✓ OK	200	10ms	UpdateItem: scorekeep-game	
DynamoDB	✓ OK	200	46ms	GetItem: scorekeep-session	
DynamoDB	✓ OK	200	39ms		

Segment details: DynamoDB

Overview | Resources | Annotations | Metadata | Exceptions | SQL

Erros, falhas e exceções

O X-Ray rastreia erros que ocorrem no código da aplicação e erros retornados por serviços subsequentes. Os erros são categorizados da seguinte forma.

- **Error**: erros de cliente (erros da série 400)
- **Fault**: erros de servidor (erros da série 500)
- **Throttle**: erros de controle de utilização (429 Solicitações em excesso)

Quando ocorre uma exceção enquanto a aplicação está atendendo a uma solicitação, o X-Ray SDK registra os detalhes da exceção, incluindo o rastreamento da pilha, se disponível. Você pode visualizar exceções em [Detalhes do segmento](#) no console do X-Ray.

Segurança em AWS X-Ray

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isto como segurança da nuvem e segurança na nuvem.

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que é executada Serviços da AWS no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. A eficácia da nossa segurança é regularmente testada e verificada por auditores de terceiros como parte dos [Programas de conformidade da AWS](#). Para saber mais sobre os programas de conformidade que se aplicam ao X-Ray, consulte [Serviços da Serviços da AWS no escopo por programa de conformidade](#).
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS service (Serviço da AWS) que você usa. Também existe a responsabilidade por outros fatores, incluindo a confidencialidade de dados, os requisitos da organização e as leis e regulamentos aplicáveis.

Esta documentação ajudará você a entender como aplicar o Modelo de Responsabilidade Compartilhada ao usar o X-Ray. Os tópicos a seguir mostram como configurar o X-Ray para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros Serviços da AWS que possam ajudá-lo a monitorar e proteger seus recursos de X-Ray.

Tópicos

- [Proteção de dados no AWS X-Ray](#)
- [Gerenciamento de identidade e acesso para AWS X-Ray](#)
- [Validação de conformidade do AWS X-Ray](#)
- [Resiliência no AWS X-Ray](#)
- [Segurança da infraestrutura no AWS X-Ray](#)

Proteção de dados no AWS X-Ray

AWS X-RayO sempre criptografa rastreamentos e dados relacionados em repouso. Quando precisar auditar e desabilitar as chaves de criptografia devido a exigências internas ou de conformidade, você

poderá configurar o X-Ray para usar uma chave do AWS Key Management Service (AWS KMS) para criptografar dados.

O X-Ray fornece uma Chave gerenciada pela AWS chamada `aws/xray`. Use essa chave quando quiser apenas [auditar o uso da chave no AWS CloudTrail](#) e não precisar gerenciar a chave em si. Quando precisar gerenciar o acesso à chave ou configurar a alternância de chaves, você poderá [criar uma chave gerenciada pelo cliente](#).

Quando você altera as configurações de criptografia, o X-Ray leva algum tempo para gerar e propagar as chaves de dados. Embora a nova chave esteja sendo processada, o X-Ray poderá criptografar os dados com a combinação de configurações novas e antigas. Os dados existentes não são criptografados novamente quando você altera as configurações de criptografia.

 Note

O AWS KMS incorre em custos quando o X-Ray usa uma chave do KMS para criptografar ou descriptografar dados de rastreamento.

- Criptografia padrão: gratuita.
- Chave gerenciada pela AWS: pague pelo uso da chave.
- Chave gerenciada pelo cliente: pague pelo armazenamento e uso da chave.

Consulte [AWS Key Management Service Pricing](#) para obter detalhes.

 Note

As notificações de insights do X-Ray enviam eventos para o Amazon EventBridge, que no momento não permite chaves gerenciadas pelo cliente. Para obter mais informações, consulte [Data Protection in Amazon EventBridge](#).

Você deve ter acesso no nível de usuário a uma chave gerenciada pelo cliente para configurar o X-Ray para usá-la e, em seguida, visualizar os rastreamentos criptografados. Consulte [Permissões de usuário para criptografia](#) para obter mais informações.

CloudWatch console

Como configurar o X-Ray para usar uma chave do KMS para criptografia utilizando o console do CloudWatch

1. Faça login no Console de gerenciamento da AWS e abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Configurações no painel de navegação à esquerda.
3. Escolha Ver configurações em Criptografia na seção Rastreamentos do X-Ray.
4. Escolha Editar na seção Configuração da criptografia.
5. Escolha Usar uma chave do KMS.
6. Escolha uma chave de acesso no menu suspenso:
 - aws/xray: use a Chave gerenciada pela AWS.
 - Alias de chave: use uma chave gerenciada pelo cliente na sua conta.
 - Inserir um ARN de chave manualmente: use uma chave gerenciada pelo cliente em uma conta diferente. Insira o nome de recurso da Amazon (ARN) completo da chave no campo exibido.
7. Escolha Atualizar criptografia.

X-Ray console

Como configurar o X-Ray para usar uma chave do KMS para criptografia usando o console do X-Ray

1. Abra o [console do X-Ray](#).
2. Escolha Encryption.
3. Escolha Usar uma chave do KMS.
4. Escolha uma chave de acesso no menu suspenso:
 - aws/xray: use a Chave gerenciada pela AWS.
 - Alias de chave: use uma chave gerenciada pelo cliente na sua conta.
 - Inserir um ARN de chave manualmente: use uma chave gerenciada pelo cliente em uma conta diferente. Insira o nome de recurso da Amazon (ARN) completo da chave no campo exibido.

5. Escolha Aplicar.

Note

O X-Ray não aceita chaves do KMS assimétricas.

Se o X-Ray não conseguir acessar sua chave de criptografia, ele interromperá o armazenamento de dados. Isso poderá acontecer se o usuário perder o acesso à chave do KMS ou se você desabilitar uma chave em uso no momento. Quando isso acontece, o X-Ray exibe uma notificação na barra de navegação.

Para definir as configurações de criptografia com a API do X-Ray, consulte [Definir configurações de amostragem, grupos e criptografia com a API do AWS X-Ray](#).

Gerenciamento de identidade e acesso para AWS X-Ray

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (fazer login) e autorizado (ter permissões) para usar os recursos do X-Ray. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- [Público](#)
- [Autenticação com identidades](#)
- [Gerenciar o acesso usando políticas](#)
- [Como o AWS X-Ray funciona com o IAM](#)
- [AWS X-Ray exemplos de políticas baseadas em identidade](#)
- [Solução de problemas de identidade e acesso do AWS X-Ray](#)

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao administrador caso você não consiga acessar os recursos (consulte [Solução de problemas de identidade e acesso do AWS X-Ray](#)).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte [Como o AWS X-Ray funciona com o IAM](#)).
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte [AWS X-Ray exemplos de políticas baseadas em identidade](#)).

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como AWS IAM Identity Center (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte [Como fazer login em sua Conta da AWS](#) no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte [AWS Signature Version 4 para solicitações de API](#) no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em [Tarefas que exigem credenciais de usuário-raiz](#) no Guia do usuário do IAM.

Usuários e grupos do IAM

[Usuário do IAM](#) é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte [Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias](#) no Guia do usuário do IAM.

Um [grupo do IAM](#) especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte [Casos de uso para usuários do IAM](#) no Guia do usuário do IAM.

Perfis do IAM

Um [perfil do IAM](#) é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função [mudando de um usuário para uma função do IAM \(console\)](#) ou chamando uma operação de AWS API AWS CLI ou. Para obter mais informações, consulte [Métodos para assumir um perfil](#) no Manual do usuário do IAM.

As funções do IAM são úteis para acesso de usuários federados, permissões temporárias de usuários do IAM, acesso entre contas, acesso entre serviços e aplicativos executados na Amazon. EC2 Consulte mais informações em [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte [Visão geral das políticas de JSON](#) no Guia do usuário do IAM.

Usando políticas, os administradores especificam quem tem acesso ao quê definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona a funções, que os usuários podem acabar assumindo. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você pode anexar a uma identidade (usuário, grupo de usuários ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Os exemplos incluem políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário [especificar uma entidade principal](#) em uma política baseada em recursos.

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a [visão geral da lista de controle de acesso \(ACL\)](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para obter mais informações, consulte [Limites de permissões para entidades do IAM](#) no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) — Especifique as permissões máximas para uma organização ou unidade organizacional em AWS Organizations. Para obter mais informações, consulte [Políticas de controle de serviço](#) no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs) — Defina o máximo de permissões disponíveis para recursos em suas contas. Para obter mais informações, consulte [Políticas de controle de recursos \(RCPs\)](#) no Guia AWS Organizations do usuário.

- Políticas de sessão: políticas avançadas passadas como um parâmetro durante a criação de uma sessão temporária para uma função ou um usuário federado. Para obter mais informações, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte [Lógica de avaliação de políticas](#) no Guia do usuário do IAM.

Como o AWS X-Ray funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao X-Ray, você precisa saber quais recursos do IAM estão disponíveis para uso com o X-Ray. Para ter uma visão geral de como o X-Ray e outros Serviços da AWS funcionam com o IAM, consulte [Serviços da AWS que funcionam com o IAM](#) no Guia do usuário do IAM.

Você pode usar o AWS Identity and Access Management (IAM) para conceder permissões do X-Ray a usuários e computar recursos na conta. O IAM controla o acesso ao serviço do X-Ray no nível da API para impor permissões de maneira uniforme, independentemente de qual cliente (console, SDK da AWS, AWS CLI) os usuários utilizem.

Para [usar o console do X-Ray](#) a fim de visualizar os mapas de serviço e os segmentos, você só precisa de permissões de leitura. Para habilitar o acesso ao console, adicione a `AWSXrayReadOnlyAccess` [política gerenciada](#) ao usuário do IAM.

Para [desenvolvimento e testes locais](#), crie um perfil do IAM com permissões de leitura e gravação. [Assuma o perfil](#) e armazene as credenciais temporárias do perfil. É possível usar essas credenciais com o daemon do X-Ray, a AWS CLI e o SDK da AWS. Para obter mais informações, consulte [Uso de credenciais de segurança temporárias com a AWS CLI](#).

Para [implantar a aplicação instrumentada na AWS](#), crie um perfil do IAM com permissões de gravação e atribua aos recursos que estão executando a aplicação. O `AWSXRayDaemonWriteAccess` inclui a permissão para carregar rastreamentos, além de algumas permissões de leitura para permitir o uso de [regras de amostragem](#).

As políticas de leitura e gravação não incluem a permissão para definir as [configurações de chaves de criptografia](#) e as regras de amostragem. Use a `AWSXrayFullAccess` para acessar essas

configurações ou adicione as [APIs de configuração](#) a uma política personalizada. Para a criptografia e a descriptografia com uma chave gerenciada pelo cliente criada por você, também será preciso obter [permissão para usar a chave](#).

Tópicos

- [Políticas baseadas em identidade do X-Ray](#)
- [Políticas baseadas em recursos do X-Ray](#)
- [Autorização baseada em tags do X-Ray](#)
- [Executar o aplicativo localmente](#)
- [Executar a aplicação na AWS](#)
- [Permissões de usuário para criptografia](#)

Políticas baseadas em identidade do X-Ray

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. O X-Ray oferece suporte a ações, recursos e chaves de condição específicos. Para conhecer todos os elementos usados em uma política JSON, consulte [Referência de elementos de política JSON do IAM](#) no Guia do usuário do IAM.

Ações

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações de políticas no X-Ray usam o seguinte prefixo antes da ação: `xray:`. Por exemplo, para conceder a alguém permissão para recuperar detalhes de recursos de grupo com a operação de API `GetGroup` do X-Ray, inclua a ação `xray:GetGroup` na política dessa pessoa. As instruções de política devem incluir um elemento `Action` ou `NotAction`. O X-Ray define um conjunto próprio de ações que descrevem as tarefas que você pode executar com esse serviço.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": [
```

```
"xray:action1",  
"xray:action2"
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra Get, inclua a seguinte ação:

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

Para ver uma lista de ações do X-Ray, consulte [Actions Defined by AWS X-Ray](#) no Guia do usuário do IAM.

Recursos

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu [nome do recurso da Amazon \(ARN\)](#). Para ações em que não é possível usar permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

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

Você pode controlar o acesso a recursos usando uma política do IAM. Para ações que aceitam permissões em nível de recurso, você usa um nome do recurso da Amazon (ARN) para identificar o recurso ao qual a política se aplica.

Todas as ações do X-Ray podem ser usadas em uma política do IAM para conceder ou negar a usuários permissão para usar essa ação. Contudo, nem todas as [ações do X-Ray](#) aceitam permissões em nível de recurso, que possibilitam especificar os recursos nos quais uma ação pode ser realizada.

Para ações que não aceitam permissões em nível de recurso, você deve usar “*” como o recurso.

As seguintes ações do X-Ray não aceitam permissões em nível de recurso:

- `CreateGroup`
- `GetGroup`
- `UpdateGroup`

- DeleteGroup
- CreateSamplingRule
- UpdateSamplingRule
- DeleteSamplingRule

A seguir, veja um exemplo de uma política de permissões baseada em identidade para uma ação do CreateGroup. O exemplo mostra o uso de um ARN relacionado ao nome do grupo local-users com o ID exclusivo como um caractere curinga. Como o ID exclusivo é gerado quando o grupo é criado, não é possível prevê-lo na política com antecedência. Ao usar GetGroup, UpdateGroup ou DeleteGroup, você pode defini-lo como um curinga ou o exato ARN, incluindo ID.

Note

O ARN de uma regra de amostragem é definido por seu nome. Ao contrário dos ARNs do grupo, as regras de amostragem não possuem ID gerado exclusivamente.

Para ver uma lista de tipos de recurso do X-Ray e os respectivos ARNs, consulte [Resources Defined by AWS X-Ray](#) do Guia do usuário do IAM. Para saber com quais ações é possível especificar o ARN de cada recurso, consulte [Ações definidas pelo AWS X-Ray](#).

Chaves de condição

O X-Ray não fornece nenhuma chave de condição específica ao serviço, mas permite o uso de algumas chaves de condição globais. Para ver todas as chaves de condição globais da AWS, consulte [Chaves de contexto de condição globais da AWS](#) no Guia do usuário do IAM.

Exemplos

Para visualizar exemplos de políticas baseadas em identidade do X-Ray, consulte [AWS X-Ray exemplos de políticas baseadas em identidade](#).

Políticas baseadas em recursos do X-Ray

O X-Ray permite políticas baseadas em recursos para integração atual e futura de AWS service (Serviço da AWS), como o [rastreamento ativo do Amazon SNS](#). As políticas baseadas em recursos do X-Ray podem ser atualizadas por outros Console de gerenciamento da AWSs ou por meio do SDK da AWS ou da CLI. Por exemplo, o console do Amazon SNS tenta configurar automaticamente

uma política baseada em recursos para enviar rastreamentos para o X-Ray. O documento de política a seguir fornece um exemplo de configuração manual da política baseada em recursos do X-Ray.

Exemplo Exemplo de política baseada em recursos do X-Ray para rastreamento ativo do Amazon SNS

Este exemplo de documento de política especifica as permissões que o Amazon SNS precisa para enviar dados de rastreamento ao X-Ray:

```
{
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "SNSAccess",
      Effect: Allow,
      Principal: {
        Service: "sns.amazonaws.com",
      },
      Action: [
        "xray:PutTraceSegments",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets"
      ],
      Resource: "*",
      Condition: {
        StringEquals: {
          "aws:SourceAccount": "account-id"
        },
        StringLike: {
          "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
        }
      }
    }
  ]
}
```

Use a CLI para criar uma política baseada em recursos que conceda permissões ao Amazon SNS para enviar dados de rastreamento ao X-Ray:

```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
'{ "Version": "2012-10-17",      "Statement": [ { "Sid": "SNSAccess",
"Effect": "Allow", "Principal": { "Service": "sns.amazonaws.com" }, "Action":
```

```
[ "xray:PutTraceSegments", "xray:GetSamplingRules", "xray:GetSamplingTargets" ],
  "Resource": "*", "Condition": { "StringEquals": { "aws:SourceAccount": "account-id" }, "StringLike": { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name" } } } ] }
```

Para usar esses exemplos, substitua *partition*, *region*, *account-id* e *topic-name* pela sua partição, região, ID da conta e nome de tópico do Amazon SNS específicos da AWS. Para conceder permissão a todos os tópicos do Amazon SNS para enviar dados de rastreamento ao X-Ray, substitua o nome do tópico por `*`.

Autorização baseada em tags do X-Ray

Você pode anexar tags a grupos ou regras de amostragem do X-Ray ou passar tags em uma solicitação para o X-Ray. Para controlar o acesso baseado em tags, forneça informações sobre as tags no [elemento de condição](#) de uma política usando as chaves de condição `xray:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`. Para obter mais informações sobre recursos de marcação do X-Ray, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte [Gerenciar o acesso a grupos e regras de amostragem do X-Ray com base em tags](#).

Executar o aplicativo localmente

A aplicação instrumentada envia dados de rastreamento para o daemon do X-Ray. O daemon armazena em buffer documentos segmentados e os carrega em lote no serviço do X-Ray. O daemon precisa de permissões de gravação para carregar os dados de rastreamento e telemetria no serviço do X-Ray.

Ao [executar o daemon localmente](#), crie um perfil do IAM, [assuma o perfil](#) e armazene credenciais temporárias em variáveis de ambiente ou em um arquivo denominado `credentials` em uma pasta chamada `.aws` na sua pasta de usuário. Para obter mais informações, consulte [Uso de credenciais de segurança temporárias com a 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}
```

Caso você já tenha configurado credenciais a serem usadas com o SDK da AWS ou a AWS CLI, o daemon pode usá-las. Caso haja vários perfis disponíveis, o daemon usa o perfil padrão.

Executar a aplicação na AWS

Ao executar a aplicação na AWS, use um perfil para conceder permissão à instância do Amazon EC2 ou à função do Lambda que executa o daemon.

- Amazon Elastic Compute Cloud (Amazon EC2): crie um perfil do IAM e anexe-o à instância do EC2 como um [perfil de instância](#).
- Amazon Elastic Container Service (Amazon ECS): crie um perfil do IAM e anexe-a às instâncias de contêiner como um [perfil do IAM da instância de contêiner](#).
- AWS Elastic Beanstalk (Elastic Beanstalk): o Elastic Beanstalk inclui permissões do X-Ray em seu [perfil de instância padrão](#). É possível usar o perfil da instância padrão ou adicionar permissões de gravação a um perfil da instância personalizado.
- AWS Lambda (Lambda): adicione permissões de gravação ao perfil de execução de sua função.

Como criar uma função a ser usada com o X-Ray

1. Abra o [console do IAM](#).
2. Escolha Perfis.
3. Escolha Criar nova função.
4. Em Nome da função, digite **xray-application**. Escolha Próxima etapa.
5. Em Tipo de função, escolha Amazon EC2.
6. Anexe a política gerenciada a seguir para oferecer ao aplicativo acesso aos Serviços da AWS.
 - AWSXRayDaemonWriteAccess: concede permissão ao daemon do X-Ray para carregar dados de rastreamento.

Se seu aplicativo usa o SDK da AWS para acessar outros serviços, adicione políticas que concedam acesso a esses serviços.

7. Escolha Próxima etapa.
8. Selecione Criar função.

Permissões de usuário para criptografia

Por padrão, o X-Ray criptografa todos os dados de rastreamento, e é possível [configurá-lo para usar uma chave gerenciada por você](#). Se você escolher uma chave gerenciada pelo cliente do AWS Key Management Service, precisará garantir que a política de acesso da chave possibilite a concessão de permissão ao X-Ray para usá-la na criptografia. Outros usuários em sua conta também precisam acessar a chave para visualizar dados de rastreamento criptografados no console do X-Ray.

Para obter uma chave gerenciada pelo cliente, configure sua chave com uma política de acesso que permita as ações a seguir.

- O usuário que configura a chave no X-Ray deve ter permissão para chamar `kms:CreateGrant` e `kms:DescribeKey`.
- Os usuários que podem acessar os dados de rastreamento criptografados têm permissão para chamar `kms:Decrypt`.

Quando você adiciona um usuário ao grupo Usuários de chaves na seção de configuração de chaves do console do IAM, ele tem permissão para executar ambas as operações. A permissão só precisa ser definida na política de chave e, portanto, você não precisa de nenhuma permissão do em seus usuários, grupos ou funções do AWS KMS. Para obter mais informações, consulte [Usar políticas de chaves no Guia do desenvolvedor do AWS KMS](#).

Para a criptografia padrão, ou se você escolher a CMK da AWS (`aws/xray`), a permissão usará como base quem tem acesso às APIs do X-Ray. Quem tiver acesso a [PutEncryptionConfig](#), incluindo em `AWSXrayFullAccess`, poderá alterar a configuração de criptografia. Para evitar que um usuário altere a chave de criptografia, não lhe dê permissão para usar a [PutEncryptionConfig](#).

AWS X-Ray exemplos de políticas baseadas em identidade

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos do X-Ray. Eles também não podem realizar tarefas usando a AWS API Console de gerenciamento da AWS AWS CLI, ou. Um administrador deve criar as políticas do IAM que concedam aos usuários e aos perfis permissões para executar operações de API específicas nos recursos especificados que precisam. O administrador deve anexar essas políticas aos usuários ou grupos que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte [Criar políticas na guia JSON](#) no Guia do usuário do IAM.

Tópicos

- [Práticas recomendadas de política](#)
- [Usar o console do X-Ray](#)
- [Permitir que os usuários visualizem suas próprias permissões](#)
- [Gerenciar o acesso a grupos e regras de amostragem do X-Ray com base em tags](#)
- [Políticas gerenciadas do IAM para o X-Ray](#)
- [Atualizações do X-Ray para políticas AWS gerenciadas](#)
- [Especificar um recurso dentro de uma política do IAM](#)

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do X-Ray em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões para seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte [Políticas gerenciadas pela AWS](#) ou [Políticas gerenciadas pela AWS para funções de trabalho](#) no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte [Políticas e permissões no IAM](#) no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation. Para obter mais informações, consulte [Elementos da política JSON do IAM: condição](#) no Guia do usuário do IAM.

- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para obter mais informações, consulte [Validação de políticas do IAM Access Analyzer](#) no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte [Configuração de acesso à API protegido por MFA](#) no Guia do Usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.

Usar o console do X-Ray

Para acessar o AWS X-Ray console, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do X-Ray em seu Conta da AWS. Caso crie uma política baseada em identidade mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Para garantir que essas entidades ainda possam usar o console X-Ray, anexe a política `AWSXRayReadOnlyAccess` AWS gerenciada às entidades. Essa política é descrita com mais detalhes nas [Políticas gerenciadas do IAM para o X-Ray](#). Para obter mais informações, consulte [Adicionar permissões a um usuário](#) no Guia do usuário do IAM.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente às ações que correspondem à operação da API que você está tentando executar.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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": "*"
    }
  ]
}
```

Gerenciar o acesso a grupos e regras de amostragem do X-Ray com base em tags

Você pode usar condições em sua política baseada em identidade para controlar o acesso a grupos e regras de amostragem do X-Ray baseadas em tags. O exemplo de política a seguir pode ser usado para negar a um perfil de usuário as permissões para criar, excluir ou atualizar grupos com as tags `stage:prod` ou `stage:preprod`. Para obter mais informações sobre a marcação de regras de amostragem e grupos do X-Ray, consulte [Marcar grupos e regras de amostragem do X-Ray](#).

Para negar a criação de uma regra de amostragem, use `aws:RequestTag` para indicar tags que não podem ser passadas como parte de uma solicitação de criação. Para negar a atualização ou exclusão de uma regra de amostragem, use `aws:ResourceTag` para negar ações com base nas tags desses recursos.

Você pode anexar essas políticas (ou combiná-las em uma única política e, em seguida, anexar a política) aos usuários da sua conta. Para que o usuário faça alterações em um grupo ou regra de amostragem, ambos não devem ser marcados com `stage=prepod` ou `stage=prod`. A chave da tag de condição `Stage` corresponde a `Stage` e a `stage` porque os nomes das chaves de condição não fazem distinção entre maiúsculas e minúsculas. Para obter mais informações sobre o uso de bloco de condição, consulte [Elementos de política JSON do IAM: Condition](#) no Guia do usuário do IAM.

Um usuário com um perfil que tenha a política anexada a seguir não pode adicionar a tag `role:admin` aos recursos e não pode remover tags de um recurso `role:admin` associado a ela.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllXRay",
      "Effect": "Allow",
      "Action": "xray:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyRequestTagAdmin",
      "Effect": "Deny",
      "Action": "xray:TagResource",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/role": "admin"
        }
      }
    },
    {
      "Sid": "DenyResourceTagAdmin",
      "Effect": "Deny",
```

```

        "Action": "xray:UntagResource",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/role": "admin"
            }
        }
    }
]
}

```

Políticas gerenciadas do IAM para o X-Ray

Para facilitar a concessão de permissões, o IAM permite políticas gerenciadas para cada serviço. Um serviço pode atualizar essas políticas gerenciadas com novas permissões quando lança novas APIs. AWS X-Ray fornece políticas gerenciadas para casos de uso somente para leitura, somente gravação e administrador.

- **AWSXrayReadOnlyAccess**— Leia as permissões para usar o console do X-Ray ou o AWS SDK para obter dados de rastreamento, mapas de rastreamento, insights e configuração do X-Ray da API X-Ray. AWS CLI Inclui o Observability Access Manager (OAM) `oam:ListSinks` e `oam:ListAttachedSinks` permissões para permitir que o console visualize traços compartilhados das contas de origem como parte da observabilidade [CloudWatch entre](#) contas. As ações `BatchGetTraceSummaryById` e `GetDistinctTraceGraphs` da API não devem ser chamadas pelo seu código e não estão incluídas no AWS CLI AWS SDKs e.

```

{
  "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**— Permissões de gravação para usar o daemon X-Ray ou AWS SDK para carregar documentos de segmentos e telemetria para a API X-Ray. AWS CLI Inclui permissões de leitura para obter [regras de amostragem](#) e relatar resultados de amostragem.

JSON

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],

```



```

        "Resource": [
            "*"
        ]
    }
]
}

```

- **AWSXrayCrossAccountSharingConfiguration**: concede permissões para criar, gerenciar e visualizar links do Observability Access Manager e compartilhar recursos do X-Ray entre contas. Usado para permitir a [observabilidade CloudWatch entre contas](#) entre contas de origem e de monitoramento.

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**— Permissão para usar todo o X-Ray APIs, incluindo permissões de leitura, permissões de gravação e permissão para definir configurações de chave de criptografia e regras de amostragem. Inclui o Observability Access Manager (OAM) `oam:ListSinks` e `oam:ListAttachedSinks` permissões para permitir que o console visualize traços compartilhados das contas de origem como parte da observabilidade [CloudWatch entre](#) contas.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:*",
        "oam:ListSinks"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "oam:ListAttachedLinks"
      ],
      "Resource": "arn:aws:oam:*:*:sink/*"
    }
  ]
}

```

Como adicionar uma política gerenciada a um usuário, grupo ou perfil do IAM

1. Abra o [console do IAM](#).
2. Abra o perfil associado ao perfil de instância, um usuário do IAM ou um grupo do IAM.

3. Em Permissions (Permissões), anexe a política gerenciada.

Atualizações do X-Ray para políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do X-Ray desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações realizadas nesta página, inscreva-se no feed RSS na página [Histórico de documentos](#).

Alteração	Descrição	Data
Políticas gerenciadas pelo IAM para o X-Ray : adição da nova política AWSXrayCrossAccountSharingConfiguration e atualização das políticas AWSXrayReadOnlyAccess e AWSXrayFullAccess .	O X-Ray adicionou permissões do Observability Access Manager (OAM) <code>oam:ListSinks</code> e <code>oam:ListAttachedSinks</code> a essas políticas para permitir que o console visualize traços compartilhados das contas de origem como parte da observabilidade CloudWatch entre contas.	27 de novembro de 2022
Políticas gerenciadas pelo IAM para X-Ray : atualização da política AWSXrayReadOnlyAccess .	O X-Ray adicionou uma ação de API, <code>ListResourcePolicies</code> .	15 de novembro de 2022
Usar o console do X-Ray : atualização da política AWSXrayReadOnlyAccess	O X-Ray adicionou duas novas ações de API, <code>BatchGetTraceSummaryById</code> e <code>GetDistinctTraceGraphs</code> . Essas ações não devem ser chamadas pelo código. Portanto, essas ações de API	11 de novembro de 2022

Alteração	Descrição	Data
	não estão incluídas no AWS CLI AWS SDKs e.	

Especificar um recurso dentro de uma política do IAM

Você pode controlar o acesso a recursos usando uma política do IAM. Para ações que aceitam permissões em nível de recurso, você usa um nome do recurso da Amazon (ARN) para identificar o recurso ao qual a política se aplica.

Todas as ações do X-Ray podem ser usadas em uma política do IAM para conceder ou negar a usuários permissão para usar essa ação. Contudo, nem todas as [ações do X-Ray](#) aceitam permissões em nível de recurso, que possibilitam especificar os recursos nos quais uma ação pode ser realizada.

Para ações que não aceitam permissões em nível de recurso, você deve usar “*” como o recurso.

As seguintes ações do X-Ray não aceitam permissões em nível de recurso:

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

A seguir, veja um exemplo de uma política de permissões baseada em identidade para uma ação do CreateGroup. O exemplo mostra o uso de um ARN relacionado ao nome do grupo `local-users` com o ID exclusivo como um caractere curinga. Como o ID exclusivo é gerado quando o grupo é criado, não é possível prevê-lo na política com antecedência. Ao usar GetGroup, UpdateGroup ou DeleteGroup, você pode defini-lo como um curinga ou o exato ARN, incluindo ID.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:CreateGroup"
      ],
      "Resource": [
        "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
      ]
    }
  ]
}
```

A seguir, veja um exemplo de uma política de permissões baseada em identidade para uma ação do `CreateSamplingRule`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:CreateSamplingRule"
      ],
      "Resource": [
        "arn:aws:xray:eu-west-1:123456789012:sampling-rule/base-scorekeep"
      ]
    }
  ]
}
```

Note

O ARN de uma regra de amostragem é definido por seu nome. Diferentemente do grupo ARNs, as regras de amostragem não têm um ID gerado exclusivamente.

Solução de problemas de identidade e acesso do AWS X-Ray

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que podem ser encontrados ao trabalhar com o X-Ray e o IAM.

Tópicos

- [Não tenho autorização para executar uma ação no X-Ray](#)
- [Não estou autorizado a executar iam:PassRole](#)
- [Sou administrador e desejo permitir que outras pessoas tenham acesso ao X-Ray](#)
- [Quero permitir que pessoas de fora da minha Conta da AWS acessem meus recursos do X-Ray](#)

Não tenho autorização para executar uma ação no X-Ray

Se o Console de gerenciamento da AWS informar que você não está autorizado a executar uma ação, você deverá entrar em contato com o administrador para obter assistência. Seu administrador é a pessoa que forneceu a você suas credenciais de início de sessão.

O erro de exemplo a seguir ocorre quando o usuário mateojackson tenta usar o console para visualizar detalhes de uma regra de amostragem e não tem permissões `xray:GetSamplingRules`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
perform: xray:GetSamplingRules on resource: arn:${Partition}:xray:${Region}:
${Account}:sampling-rule/${SamplingRuleName}
```

Nesse caso, Mateo pede ao administrador para atualizar suas políticas a fim de obter acesso ao recurso de regra de amostragem usando a ação `xray:GetSamplingRules`.

Não estou autorizado a executar iam:PassRole

Se você receber uma mensagem de erro informando que não tem autorização para executar a ação `iam:PassRole`, suas políticas deverão ser atualizadas para permitir que você passe um perfil para o X-Ray.

Alguns Serviços da AWS permitem que você passe um perfil existente para o serviço, em vez de criar um perfil de serviço ou perfil vinculado ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro de exemplo a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no X-Ray. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

Sou administrador e desejo permitir que outras pessoas tenham acesso ao X-Ray

Para permitir que outros usuários acessem o X-Ray, conceda permissão para pessoas ou aplicativos que precisam do acesso. Se você estiver usando AWS IAM Identity Center para gerenciar pessoas e aplicativos, você atribui conjuntos de permissões a usuários ou grupos para definir seu nível de acesso. Os conjuntos de permissões criam e atribuem automaticamente políticas do IAM aos perfis do IAM associados à pessoa ou aplicação. Para ter mais informações, consulte [Conjuntos de permissões](#) no Guia do usuário do AWS IAM Identity Center.

Caso não esteja usando o Centro de Identidade do IAM, crie entidades do IAM (usuários ou perfis) para pessoas ou aplicativos que precisam do acesso. Você deve anexar uma política à entidade que concede a eles as permissões corretas no X-Ray. Depois que as permissões forem concedidas, forneça as credenciais ao usuário ou desenvolvedor do aplicativo. Eles usarão essas credenciais para acessar AWS. Para saber mais sobre como criar grupos, políticas, permissões e usuários do IAM, consulte [Criar um grupo de administradores usando o console](#) e [Permissões e políticas](#) no Manual do usuário do IAM.

Quero permitir que pessoas de fora da minha Conta da AWS acessem meus recursos do X-Ray

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para

serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o X-Ray comporta esses recursos, consulte [Como o AWS X-Ray funciona com o IAM](#).
- Para saber como conceder acesso a seus recursos em todas as Contas da AWS pertencentes a você, consulte [Fornecimento de acesso a um usuário do IAM em outra Conta da AWS pertencente a você](#) no Guia de usuário do IAM.
- Para saber como conceder acesso a seus recursos para Contas da AWS de terceiros, consulte [Fornecimento de acesso a Contas da AWS pertencentes a terceiros](#) no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte [Conceder acesso a usuários autenticados externamente \(federação de identidades\)](#) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Registrar em log e monitorar no AWS X-Ray

O monitoramento é uma parte importante da manutenção da confiabilidade, da disponibilidade e da performance das suas soluções da AWS. Você deve coletar dados de monitoramento de todas as partes de sua solução da AWS para depurar uma falha de vários pontos com maior facilidade, caso ocorra. A AWS fornece várias ferramentas para monitorar recursos do X-Ray e responder a possíveis incidentes:

AWS CloudTrailLogs do

O AWS X-Ray se integra ao AWS CloudTrail para registrar as ações de API realizadas por um usuário, um perfil ou um serviço da AWS no X-Ray. Você pode usar o CloudTrail para monitorar solicitações de API do X-Ray em tempo real e armazenar logs no Amazon S3, Amazon CloudWatch Logs e Amazon CloudWatch Events. Para obter mais informações, consulte [Registro em log de chamadas de API do X-Ray com o AWS CloudTrail](#).

AWS Config Rastrear

O AWS X-Ray se integra ao AWS Config para registrar as alterações de configuração feitas nos recursos de criptografia do X-Ray. Você pode usar o AWS Config para inventariar os recursos

de criptografia do X-Ray, auditar o histórico de configuração do X-Ray e enviar notificações com base nas alterações de recursos. Para obter mais informações, consulte [Rastrear as alterações da configuração de criptografia do X-Ray com o AWS Config](#).

Monitoramento do Amazon CloudWatch

Você pode usar o X-Ray SDK para Java para publicar métricas sem amostragem do Amazon CloudWatch usando os segmentos do X-Ray coletados. Essas métricas são derivadas da hora de início e término do segmento e dos sinalizadores de status de erro, falha e limitação. Use essas métricas de rastreamento para expor novas tentativas e problemas de dependência dentro de subsegmentos. Para obter mais informações, consulte [AWS X-Ray métricas para o X-Ray SDK for Java](#).

Validação de conformidade do AWS X-Ray

Para saber se um AWS service (Serviço da AWS) está no escopo de programas de conformidade específicos, consulte [Serviços da AWS no escopo por programa de conformidade](#) e selecione o programa de conformidade em que você está interessado. Para obter informações gerais, consulte [Programas de Conformidade da AWS](#).

É possível baixar relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte [Baixar relatórios no AWS Artifact](#).

Sua responsabilidade de conformidade ao usar o Serviços da AWS é determinada pela confidencialidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelos regulamentos e leis aplicáveis. Para ter mais informações sobre sua responsabilidade pela conformidade ao usar Serviços da AWS, consulte a [documentação da AWS sobre segurança](#).

Resiliência no AWS X-Ray

A infraestrutura global da AWS se baseia em Regiões da AWS e zonas de disponibilidade. A Regiões da AWS oferece várias zonas de disponibilidade separadas e isoladas fisicamente que são conectadas com baixa latência, altas taxas de throughput e em redes altamente redundantes. Com as Zonas de Disponibilidade, é possível projetar e operar aplicações e bancos de dados que executem o failover automaticamente entre as Zonas de Disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre Regiões da AWS e zonas de disponibilidade, consulte [Infraestrutura global da AWS](#).

Segurança da infraestrutura no AWS X-Ray

Por ser um serviço gerenciado, o AWS X-Ray é protegido pela segurança da rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte [Segurança na Nuvem AWS](#). Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança de infraestrutura, consulte [Proteção de infraestrutura](#) em Pilar segurança: AWS Well-Architected Framework.

Você usa chamadas de API publicadas pela AWS para acessar o X-Ray por meio da rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Usar o AWS X-Ray com endpoints da VPC

Se você usar o Amazon Virtual Private Cloud (Amazon VPC) para hospedar os recursos da AWS, poderá estabelecer uma conexão privada entre sua VPC e o X-Ray. Isso permite que os recursos do Amazon VPC se comuniquem com o serviço X-Ray sem passar pela internet pública.

A Amazon VPC é um AWS service (Serviço da AWS) que pode ser utilizado para iniciar os recursos da AWS em uma rede virtual definida por você. Com a VPC, você tem controle sobre as configurações de rede, como o intervalo de endereços IP, sub-redes, tabelas de rotas e gateways de rede. Para conectar a VPC ao X-Ray, defina uma [interface do endpoint da VPC](#). O endpoint fornece uma conectividade confiável e escalável ao X-Ray sem a necessidade de um gateway da Internet, de uma instância de conversão de endereços de rede (NAT) ou de uma conexão VPN. Para obter mais informações, consulte [O que é a Amazon VPC?](#) no Guia do usuário da Amazon VPC.

Os endpoints da VPC de interface são viabilizados pelo AWS PrivateLink, uma tecnologia da AWS que permite a comunicação privada entre os Serviços da AWS usando uma interface de rede elástica com endereços IP privados. Para obter mais informações, consulte a publicação de blog [Apresentação do AWS PrivateLink para Serviços da AWS](#) e [Conceitos básicos](#) do Manual do usuário do Amazon VPC.

Para garantir que você possa criar um endpoint da VPC para X-Ray na Região da AWS escolhida, consulte [Regiões com suporte](#).

Criar um endpoint da VPC para o X-Ray

Para começar a usar o X-Ray com a VPC, crie um endpoint da VPC para o X-Ray.

1. Abra o console da Amazon VPC, em <https://console.aws.amazon.com/vpc/>.
2. Navegue até Endpoints no painel de navegação e escolha Criar endpoint.
3. Pesquise e selecione o nome do serviço AWS X-Ray: `com.amazonaws.region.xray`.

Service category ☒ AWS services
☐ Find service by name
☐ Your AWS Marketplace services

Service Name `com.amazonaws.us-west-2.xray` ⓘ

Filter by attributes or search by keyword			
	Service Name	Owner	Type
☐	com.amazonaws.us-west-2.transfer.server	amazon	Interface
☐	com.amazonaws.us-west-2.workspaces	amazon	Interface
☒	com.amazonaws.us-west-2.xray	amazon	Interface

4. Selecione a VPC que você deseja e escolha uma sub-rede na VPC para usar o endpoint de interface. Uma interface de rede de endpoint é criada na sub-rede selecionada. Você pode especificar mais de uma sub-rede em diferentes zonas de disponibilidade (conforme compatível por serviço) para ajudar a garantir que seu endpoint de interface seja resiliente a falhas da zona de disponibilidade. Se você fizer isso, uma interface de rede é criada em cada sub-rede que você habilitar.

VPC* `vpc-4f6e3a37` ↕ ⓘ

Subnets `subnet-40d87938` ⓘ

Availability Zone	Subnet ID
☒ us-west-2a (usw2-az1)	subnet-40d87938
☐ us-west-2b (usw2-az2)	subnet-ff4281b5
☐ us-west-2c (usw2-az3)	subnet-d14bfb8c
☐ us-west-2d (usw2-az4)	subnet-1faf8734

5. (Opcional) O DNS privado é habilitado por padrão para o endpoint, para que você possa fazer solicitações ao X-Ray usando o nome de host do DNS padrão. Você pode optar por desabilitá-lo.
6. Especifique os grupos de segurança a serem associados à interface de rede do endpoint.

Security group **sg-d4f14ff4** [Create a new security group](#) ⓘ

Select security groups ▲

1 to 5 of 5

☐	Group ID	Group Name	VPC ID		Description	Owner ID
☐	sg-0683c...	ssh-http	vpc-4f6e3a37	EC2-VPC	launch-wizar...	979300271395
☐	sg-0774...	awseb-e-7xv5...	vpc-4f6e3a37	EC2-VPC	SecurityGrou...	979300271395
☐	sg-0a46...	launch-wizard-1	vpc-4f6e3a37	EC2-VPC	launch-wizar...	979300271395
☐	sg-0d62...	awseb-e-7xv5...	vpc-4f6e3a37	EC2-VPC	Elastic Beans...	979300271395
☒	sg-d4f14...	default	vpc-4f6e3a37	EC2-VPC	default VPC s...	979300271395

[Close](#)

7. (Opcional) Especifique uma política personalizada para controlar as permissões de acesso ao serviço X-Ray. Por padrão, o acesso total é permitido.

Controlar o acesso ao endpoint da VPC do X-Ray

Uma política de endpoint da VPC é uma política de recursos do IAM que você anexa a um endpoint quando cria ou modifica o endpoint. Se você não associar uma política ao criar um endpoint, a Amazon VPC associará uma política padrão que permita o acesso total ao serviço. Uma política de endpoint não substitui políticas de usuário do IAM nem políticas de serviço específicas. É uma política separada para controlar o acesso do endpoint ao serviço especificado. Políticas de endpoint devem ser gravadas em formato JSON. Para obter mais informações, consulte [Controlar o acesso a serviços com VPC endpoints](#) no Manual do usuário da Amazon VPC.

A política de endpoint da VPC permite que você controle permissões para várias ações do X-Ray. Por exemplo, você pode criar uma política para permitir somente PutTraceSegment e negar todas as outras ações. Isso restringe workloads e serviços na VPC para que enviem somente dados de rastreamento para o X-Ray e neguem qualquer outra ação, como recuperar dados, alterar a configuração de criptografia ou criar e atualizar grupos.

Veja a seguir um exemplo de uma política de endpoint para o X-Ray. Essa política permite que os usuários que se conectam ao X-Ray por meio da VPC enviem dados de segmento ao X-Ray e também os impede de executar outras ações do X-Ray.

```
{
  "Statement": [
    {
      "Sid": "Allow PutTraceSegments",
      "Principal": "*",
      "Action": [
        "xray:PutTraceSegments"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Como editar a política de endpoint da VPC para o X-Ray

1. Abra o console da Amazon VPC, em <https://console.aws.amazon.com/vpc/>.
2. No painel de navegação, escolha Endpoints.
3. Se você ainda não criou o endpoint para o X-Ray, siga as etapas em [Criar um endpoint da VPC para o X-Ray](#).
4. Selecione o endpoint com `amazonaws.region.xray` e escolha a guia Política.
5. Escolha Edit Policy (Editar política) e faça as alterações.

Regiões com suporte

No momento, o X-Ray comporta endpoints da VPC nas seguintes Regiões da AWS:

- Leste dos EUA (Ohio)
- Leste dos EUA (Norte da Virgínia)
- Oeste dos EUA (Norte da Califórnia)
- Oeste dos EUA (Oregon)
- África (Cidade do Cabo)
- Ásia-Pacífico (Hong Kong)
- Ásia-Pacífico (Mumbai)

- Ásia-Pacífico (Osaka)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- Ásia-Pacífico (Tóquio)
- Canadá (Central)
- Europa (Frankfurt)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milão)
- Europa (Paris)
- Europa (Estocolmo)
- Oriente Médio (Bahrein)
- South America (São Paulo)
- AWS GovCloud (Leste dos EUA)
- AWS GovCloud (Oeste dos EUA)

Prevenção contra o ataque do “substituto confuso” em todos os serviços

“Confused deputy” é um problema de segurança no qual uma entidade sem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Na AWS, a personificação entre serviços pode resultar no problema do ‘confused deputy’. A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta.

Recomendamos o uso das chaves de contexto de condição global [aws:SourceArn](#), [aws:SourceAccount](#), [aws:SourceOrgID](#) e [aws:SourceOrgPaths](#) nas políticas de recursos para limitar as permissões para o recurso que o xraylong dá a outro serviço. Use `aws:SourceArn` se quiser associar apenas um recurso ao acesso entre serviços. Use `aws:SourceAccount`

se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços. Use `aws:SourceOrgID` se quiser permitir que qualquer recurso de qualquer conta de uma organização seja associado ao uso entre serviços. Use `aws:SourceOrgPaths` se quiser associar qualquer recurso das contas em um caminho do AWS Organizations seja associado ao uso entre serviços. Para obter mais informações sobre como usar e entender os caminhos, consulte [Entender o caminho da entidade do AWS Organizations](#).

A maneira mais eficaz de se proteger contra o problema do substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com caracteres curinga (*) para as partes desconhecidas do ARN. Por exemplo, `arn:aws:servicename*:123456789012:*`.

Se o valor do `aws:SourceArn` não contiver o ID da conta, como um ARN de bucket do Amazon S3, você deverá usar ambos, a `aws:SourceAccount` e o `aws:SourceArn` para limitar as permissões.

Para se proteger do problema de "confused deputy" em grande escala, use a chave de contexto de condição global `aws:SourceOrgID` ou `aws:SourceOrgPaths` com o ID ou o caminho da organização do recurso nas políticas baseadas em recursos. As políticas que incluem a chave `aws:SourceOrgID` ou `aws:SourceOrgPaths` incluem automaticamente as contas corretas e você não tem que atualizar manualmente as políticas quando adiciona, remove ou move contas na organização.

O exemplo a seguir mostra como você pode usar as chaves de contexto de condição globais `aws:SourceArn` e `aws:SourceAccount` no xray para evitar o problema confused deputy.

```
{
  "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:*"  
    }  
  }  
}
```

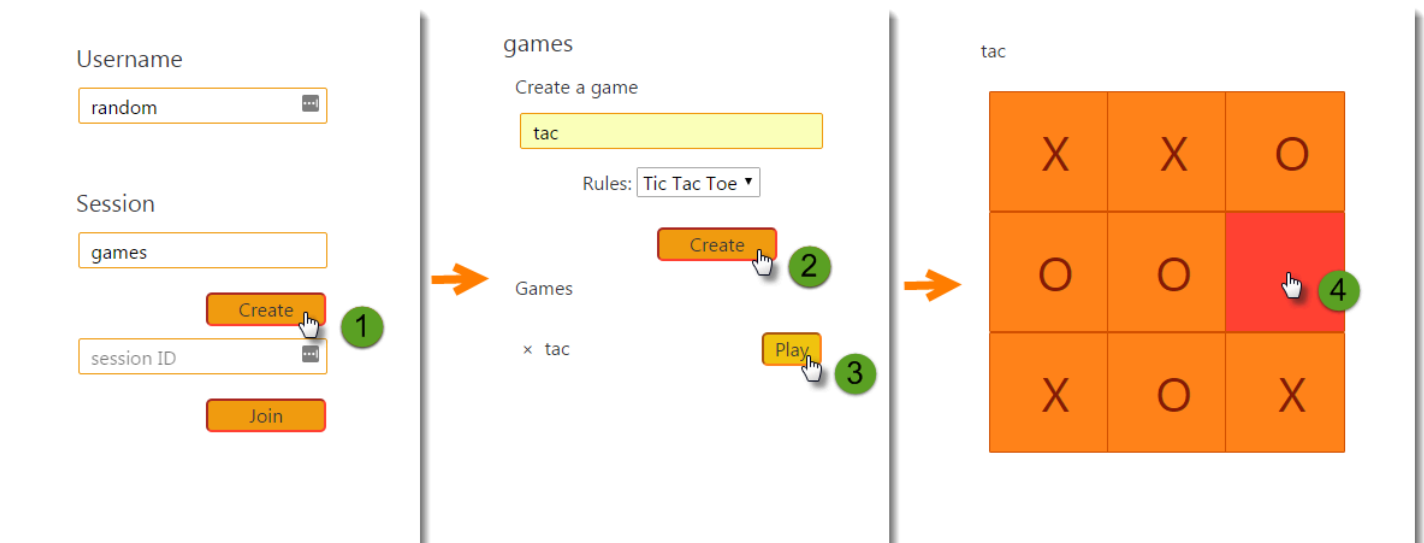

AWS X-Ray aplicação de amostra

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

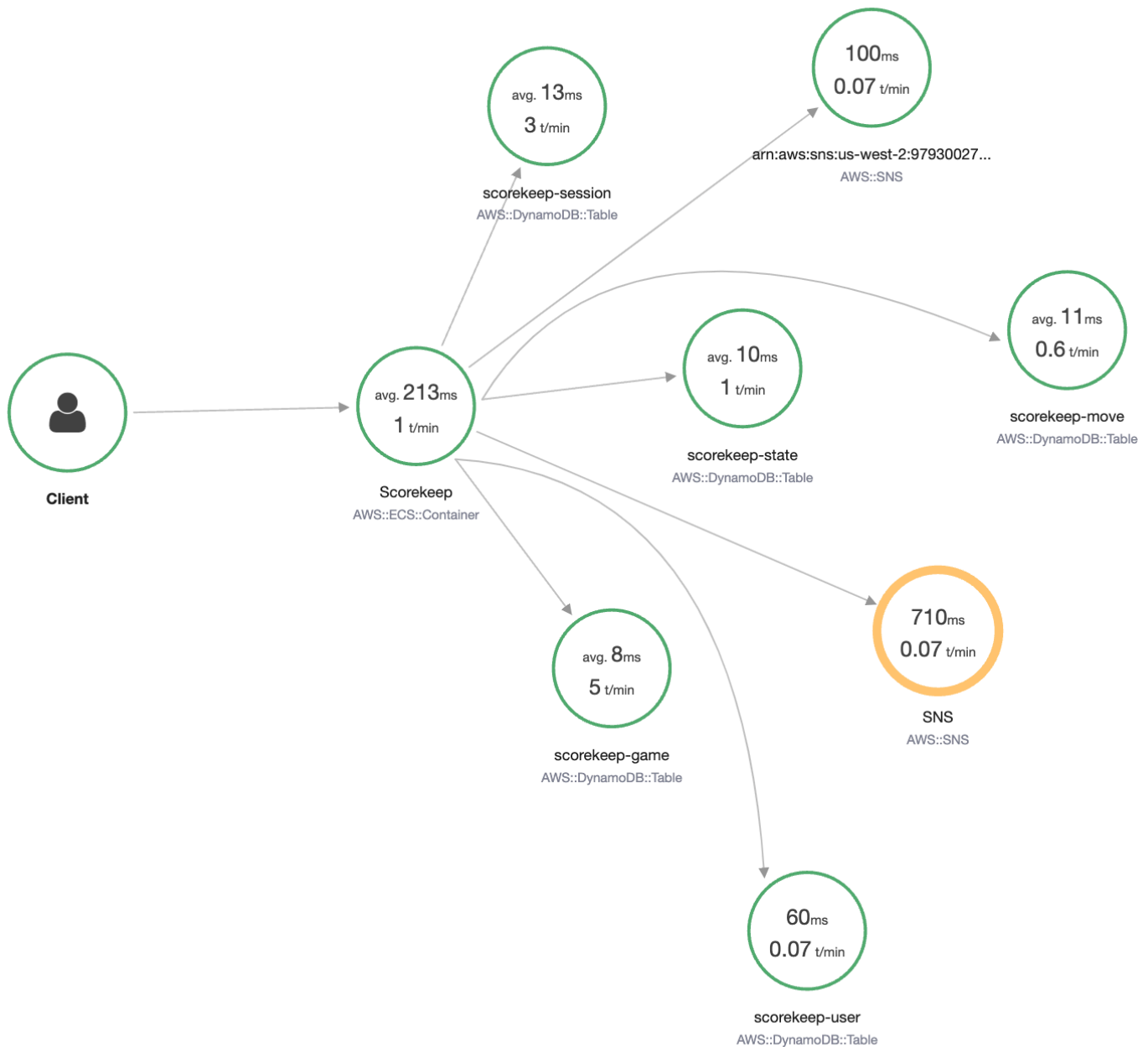
O aplicativo AWS X-Ray [eb-java-scorekeep](#) Sample, disponível em GitHub, mostra o uso do AWS X-Ray SDK para instrumentar chamadas HTTP recebidas, clientes SDK do DynamoDB e clientes HTTP. O aplicativo de amostra é usado CloudFormation para criar tabelas do DynamoDB, compilar código Java na instância e executar o daemon X-Ray sem nenhuma configuração adicional.

Consulte o [tutorial do Scorekeep](#) para começar a instalar e usar um aplicativo de amostra instrumentado, usando o Console de gerenciamento da AWS ou o AWS CLI



O exemplo inclui um aplicativo web de front-end, a API chamada e as tabelas do DynamoDB usadas para armazenar os dados. A instrumentação básica com [filtros](#), [plug-ins](#) e [clientes AWS SDK instrumentados](#) é mostrada na ramificação do projeto. `xray-gettingstarted` Essa é a

ramificação que você implanta no [tutorial de conceitos básicos](#). Como essa ramificação inclui apenas os conceitos básicos, você pode compará-la com a ramificação master para compreender rapidamente os conceitos básicos.



O aplicativo de amostra mostra a instrumentação básica nesses arquivos:

- Filtro de solicitação HTTP: [WebConfig.java](#)
- AWS Instrumentação do cliente SDK — [build.gradle](#)

[A `xray` ramificação do aplicativo inclui o uso de anotações HTTPClient, consultas SQL, subsegmentos personalizados, uma AWS Lambda função instrumentada e códigos e scripts de inicialização instrumentados.](#)

Para oferecer suporte ao login e ao AWS SDK para JavaScript uso do usuário no navegador, a `xray-cognito` filial adiciona o Amazon Cognito para oferecer suporte à autenticação e autorização do usuário. Com as credenciais recuperadas do Amazon Cognito, o aplicativo web também envia dados de rastreamento ao X-Ray para registrar as informações da solicitação do ponto de vista do cliente. O cliente do navegador aparece como seu próprio nó no mapa de rastreamento e grava informações adicionais, incluindo o URL da página que o usuário está visualizando e o ID do usuário.

Finalmente, a ramificação `xray-worker` adiciona uma função do Lambda instrumentada em Python que é executada de forma independente e processa os itens com base em uma fila do Amazon SQS. O Scorekeep adiciona um item à fila cada vez que um jogo termina. O trabalhador Lambda, acionado por CloudWatch Eventos, retira itens da fila a cada poucos minutos e os processa para armazenar registros do jogo no Amazon S3 para análise.

Tópicos

- [Conceitos básicos do aplicativo Scorekeep de exemplo](#)
- [Instrumentando manualmente os clientes do AWS SDK](#)
- [Criar subsegmentos adicionais](#)
- [Gravando anotações, metadados e usuário IDs](#)
- [Instrumentar chamadas HTTP de saída](#)
- [Instrumentação de chamadas para um banco de dados PostgreSQL](#)
- [Funções de instrumentação AWS Lambda](#)
- [Instrumentar código de inicialização](#)
- [Scripts de instrumentação](#)
- [Instrumentar o cliente do aplicativo web](#)
- [Usar clientes instrumentais em threads de operador](#)

Conceitos básicos do aplicativo Scorekeep de exemplo

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Este tutorial usa a `xray-gettingstarted` ramificação do [aplicativo de amostra Scorekeep](#), usada CloudFormation para criar e configurar os recursos que executam o aplicativo de amostra e o daemon X-Ray no Amazon ECS. O aplicativo usa a estrutura Spring para implementar uma API web JSON e AWS SDK para Java para manter os dados no Amazon DynamoDB. Um filtro de servlet no aplicativo instrumenta todas as solicitações recebidas atendidas pelo aplicativo e um manipulador de solicitações no cliente AWS SDK instrumenta chamadas downstream para o DynamoDB.

Você pode seguir este tutorial usando o Console de gerenciamento da AWS ou AWS CLI o.

Seções

- [Pré-requisitos](#)
- [Instale o aplicativo Scorekeep usando CloudFormation](#)
- [Gerar dados de rastreamento](#)
- [Veja o mapa de rastreamento no Console de gerenciamento da AWS](#)
- [Configuração de notificações do Amazon SNS](#)
- [Explorar o aplicativo de amostra](#)
- [Opcional: política de menor privilégio](#)
- [Limpeza](#)
- [Próximas etapas](#)

Pré-requisitos

Este tutorial é usado CloudFormation para criar e configurar os recursos que executam o aplicativo de amostra e o daemon X-Ray. Para instalar e executar o tutorial, você deve cumprir os seguintes pré-requisitos:

1. Se você usa um usuário do IAM com permissões limitadas, adicione as seguintes políticas de usuário no [console do IAM](#):
 - `AWSCloudFormationFullAccess`— para acessar e usar CloudFormation
 - `AmazonS3FullAccess`— para fazer upload de um arquivo de modelo para CloudFormation usar o Console de gerenciamento da AWS
 - `IAMFullAccess`— para criar as funções de EC2 instância do Amazon ECS e da Amazon
 - `AmazonEC2FullAccess`— para criar os EC2 recursos da Amazon
 - `AmazonDynamoDBFullAccess`: para criar as tabelas do DynamoDB
 - `AmazonECS_FullAccess`: para criar recursos do Amazon ECS
 - `AmazonSNSFullAccess`: para criar o tópico do Amazon SNS
 - `AWSXrayReadOnlyAccess`: para obter permissão para visualizar o mapa de serviço e os rastreamentos no console do X-Ray
2. Para executar o tutorial usando o AWS CLI, [instale a CLI](#) versão 2.7.9 ou posterior e configure [a CLI com o](#) usuário da etapa anterior. Certifique-se de que a região esteja configurada ao configurar o AWS CLI com o usuário. Se uma região não estiver configurada, você precisará anexar `--region AWS-REGION` a cada comando da CLI.
3. O [Git](#) deve estar instalado para clonar o repositório do aplicativo de exemplo.
4. Use o exemplo de código a seguir para clonar a `xray-gettingstarted` ramificação do repositório Scorekeep:

```
git clone https://github.com/aws-samples/eb-java-scorekeep.git xray-scorekeep -b  
xray-gettingstarted
```

Instale o aplicativo Scorekeep usando CloudFormation

Console de gerenciamento da AWS

Instale o aplicativo de amostra usando o Console de gerenciamento da AWS

1. Abra o [console do CloudFormation](#).
2. Escolha Criar pilha e selecione Com novos recursos no menu suspenso.
3. Na seção Specify template (Especificar modelo) escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione Escolher arquivo, navegue até a pasta `xray-scorekeep/cloudformation` que foi criada quando você clonou o repositório do Git e escolha o arquivo `cf-resources.yaml`.
5. Escolha Próximo para continuar.
6. Insira `scorekeep` na caixa de texto Nome da pilha e escolha Próximo na parte inferior da página para continuar. Observe que o restante deste tutorial pressupõe que a pilha se chame `scorekeep`.
7. Role até a parte inferior da página Configurar opções de pilha e escolha Próximo para continuar.
8. Role até a parte inferior da página de revisão, escolha a caixa de seleção reconhecendo que CloudFormation pode criar recursos do IAM com nomes personalizados e escolha Criar pilha.
9. A CloudFormation pilha agora está sendo criada. O status da pilha será `CREATE_IN_PROGRESS` por cerca de cinco minutos antes de mudar para `CREATE_COMPLETE`. O status será atualizado periodicamente ou você poderá atualizar a página.

AWS CLI

Instale o aplicativo de amostra usando o AWS CLI

1. Navegue até a pasta `cloudformation` do repositório do `xray-scorekeep` que você clonou anteriormente no tutorial:

```
cd xray-scorekeep/cloudformation/
```

2. Digite o seguinte AWS CLI comando para criar a CloudFormation pilha:

```
aws cloudformation create-stack --stack-name scorekeep --capabilities  
"CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

3. Espere até que o status da CloudFormation pilha seja `CREATE_COMPLETE`, o que levará cerca de cinco minutos. Use o AWS CLI comando a seguir para verificar o status:

```
aws cloudformation describe-stacks --stack-name scorekeep --query  
"Stacks[0].StackStatus"
```

Gerar dados de rastreamento

O aplicativo de amostra inclui um aplicativo Web front-end. Use o aplicativo web para gerar tráfego para a API e enviar dados de rastreamento para o X-Ray. Primeiro, recupere o URL do aplicativo web usando o Console de gerenciamento da AWS ou a AWS CLI:

Console de gerenciamento da AWS

Encontre o URL do aplicativo usando o Console de gerenciamento da AWS

1. Abra o [console do CloudFormation](#).
2. Escolha a pilha `scorekeep` na lista.
3. Escolha a guia Saídas na página da pilha `scorekeep` e selecione o link do URL `LoadBalancerUrl` para abrir o aplicativo web.

AWS CLI

Encontre o URL do aplicativo usando o AWS CLI

1. Use o seguinte comando para exibir o URL do aplicativo web:

```
aws cloudformation describe-stacks --stack-name scorekeep --query  
"Stacks[0].Outputs[0].OutputValue"
```

2. Copie esse URL e abra em um navegador para exibir o aplicativo web Scorekeep.

Usar o aplicativo web para gerar dados de rastreamento

1. Escolha Create para criar um usuário e uma sessão.
2. Digite um game name, defina Rules para Tic Tac Toe e, em seguida, clique em Create para criar um jogo.
3. Escolha Play para iniciar o jogo.
4. Escolha um bloco para fazer um movimento e alterar o estado do jogo.

Cada uma dessas etapas gera solicitações HTTP para a API e chamadas subsequentes para o DynamoDB para ler e gravar dados de usuário, sessão, jogo, movimento e estado.

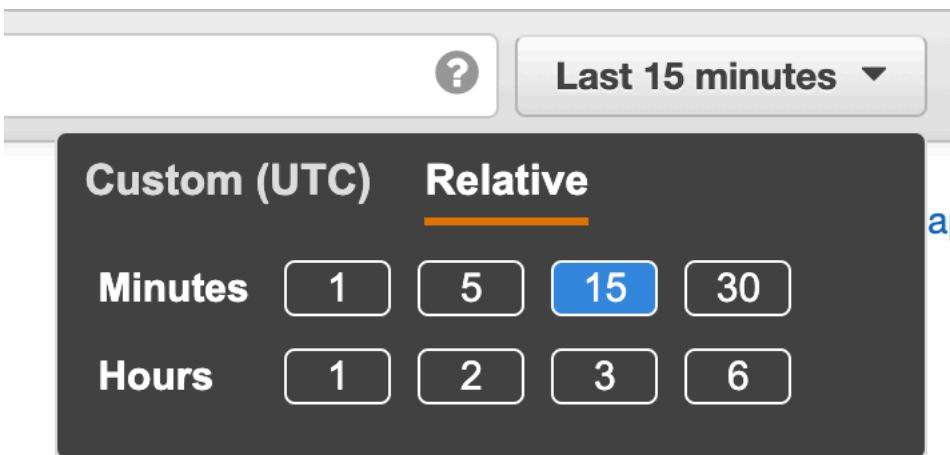
Veja o mapa de rastreamento no Console de gerenciamento da AWS

Você pode ver o mapa de rastreamento e os traços gerados pelo aplicativo de amostra no X-Ray e nos CloudWatch consoles.

X-Ray console

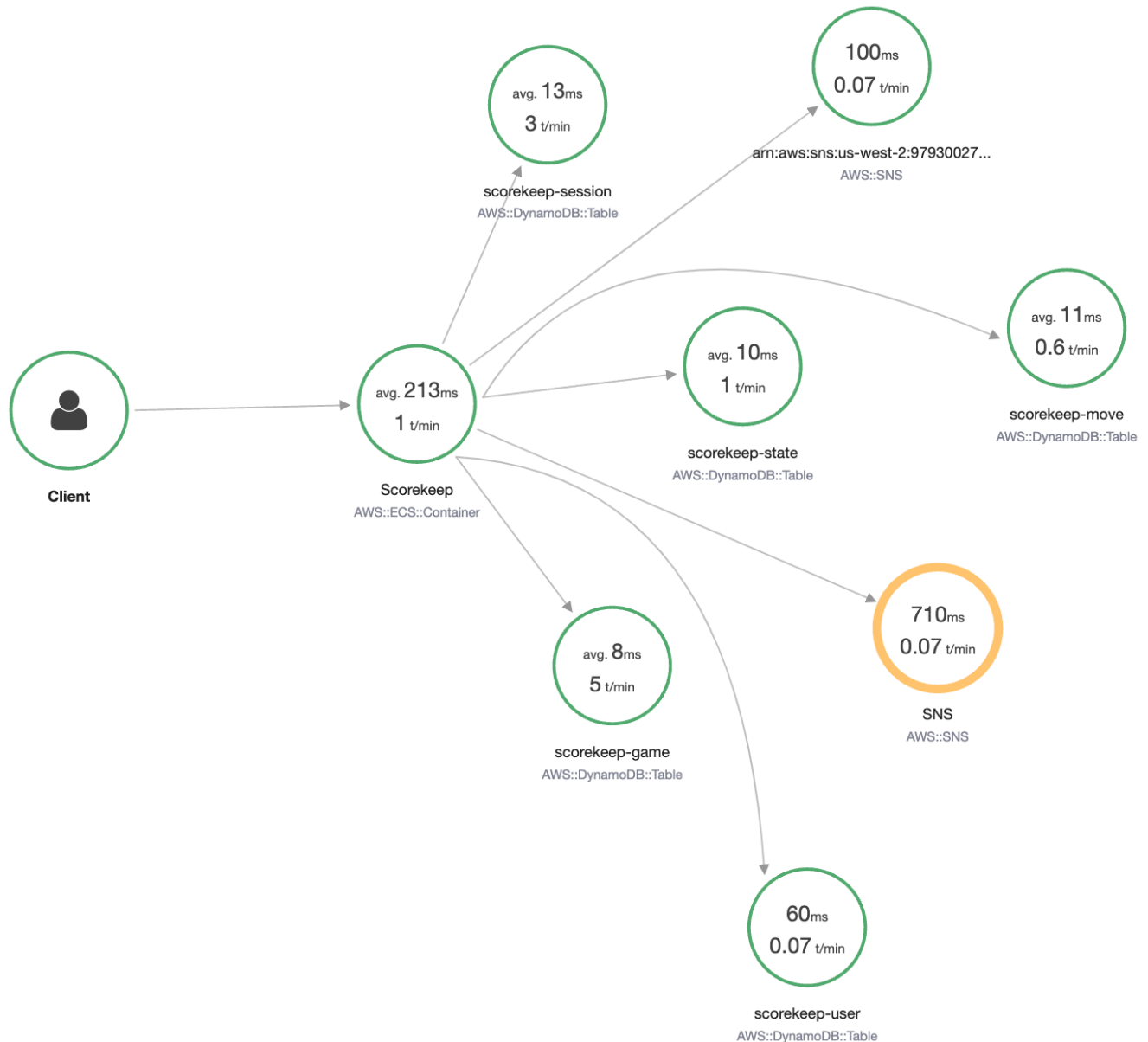
Usar o console do X-Ray

1. Abra a página do mapa de serviço do [console do X-Ray](#).
2. O console mostra uma representação do gráfico de serviço que o X-Ray gera com base nos dados de rastreamento enviados pelo aplicativo. Ajuste o período do mapa de serviço, se necessário, para garantir que ele exiba todos os rastreamentos desde que você iniciou o aplicativo web.



O mapa de serviço mostra o cliente do aplicativo web, a API em execução no Amazon ECS e cada tabela do DynamoDB que o aplicativo usa. Cada solicitação para o aplicativo, até um número máximo configurável de solicitações por segundo, é rastreada quando atinge a API, gera solicitações para serviços subsequentes e é concluída.

Você pode escolher qualquer nó no gráfico do serviço para visualizar rastreamentos de solicitações que geraram tráfego nesse nó. No momento, o nó do Amazon SNS é amarelo. Pesquise mais para descobrir o motivo.



Para encontrar a causa do erro

1. Escolha o nó chamado SNS. A página de detalhes de nó será exibida.
2. Selecione View traces (Visualizar rastreamentos) para acessar a tela Trace overview (Visão geral do rastreamento).
3. Escolha o rastreamento na Trace list. Esse rastreamento não tem um método ou URL porque, em vez de ele ser registrado em resposta a uma solicitação recebida, seu registro foi feito durante o startup.

The screenshot shows the AWS X-Ray console interface. At the top, there is a search bar with the text "service('SNS')", a help icon, and a "Last 5 Minutes" filter. Below this is the "Trace overview" section, which includes a "Group by:" dropdown set to "URL". A table displays summary statistics: a single row with a hyphen in the URL column, 1.3 sec for Avg Latency, 100.00% for % of Traces, and a response of "1 OK, 0 Throttled, 0 Errors, 0 Faults". Below the overview is the "Trace list (1)" section, which contains a table with one trace. The first row of the table has columns: ID, Age, Method, Response, Latency, URL, Client IP, and Annotations. The values for this row are: ID "...48b5a191" (highlighted with an orange box), Age "1.1 min", Method (empty), Response (empty), Latency "1.3 sec", URL (empty), Client IP (empty), and Annotations "0".

URL	Avg Latency	% of Traces	Response
-	1.3 sec	100.00%	1 OK, 0 Throttled, 0 Errors, 0 Faults

ID	Age	Method	Response	Latency	URL	Client IP	Annotations
...48b5a191	1.1 min			1.3 sec			0

4. Escolha o ícone de status de erro no segmento do Amazon SNS na parte inferior da página para abrir a página Exceções do subsegmento do SNS.

Traces > Details

Q 1-62f40175-86b347fc50bc57a992e9b835

?

↺

Timeline

Raw data

Method	Response	Duration	Age	ID
--	--	2.1 sec	8.3 min (2022-08-10 19:05:25 UTC)	1-62f40175-86b347fc50bc57a992e9b835

▼ Trace Map

Client

2.11s

1 Request

Scorekeep

AWS::EC2::Instance

728ms

1 Request

SNS

AWS::SNS

Services Icons

None

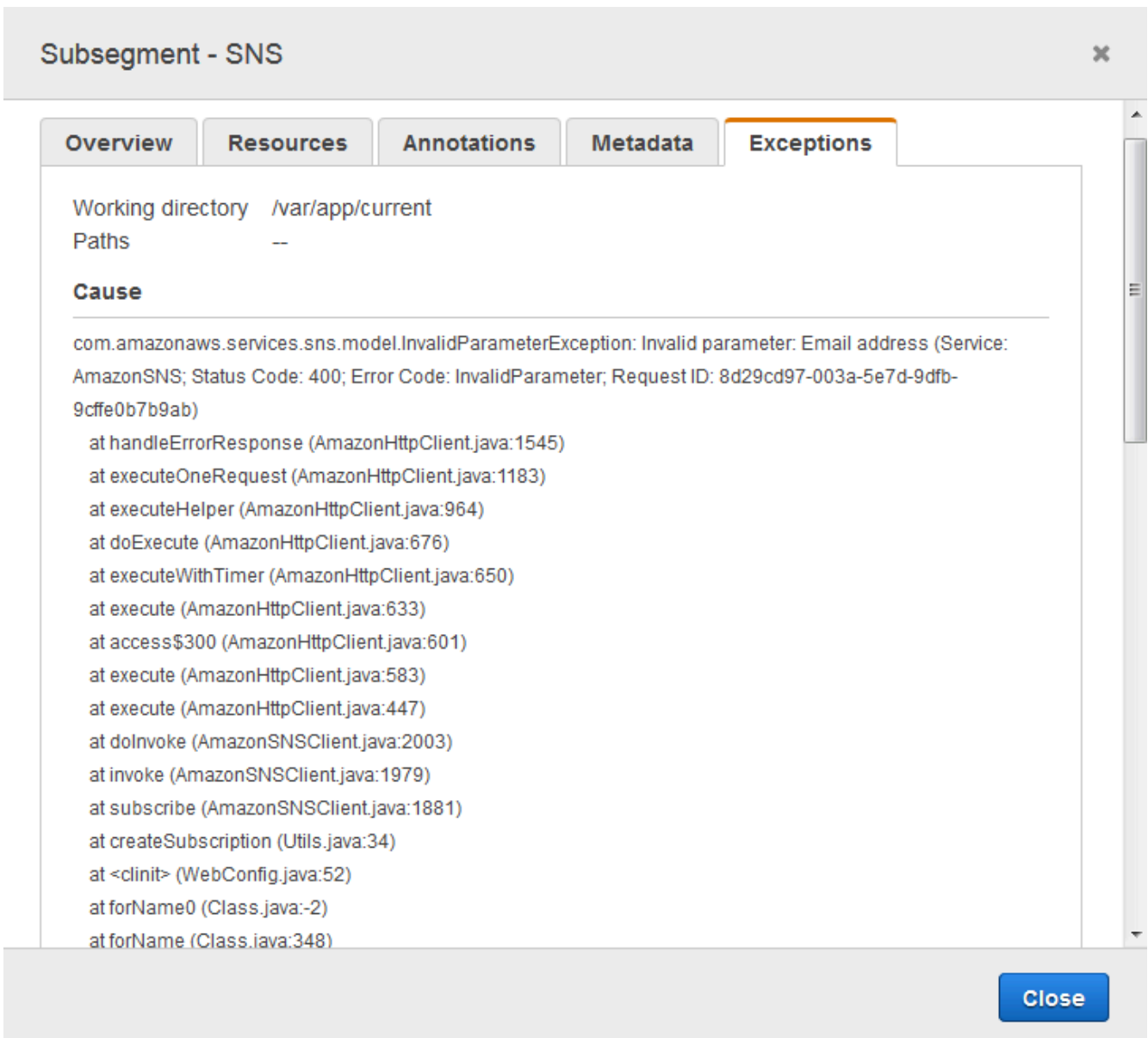
Health

Traffic

No node resizing

Name	Res.	Duration	Status	0.0ms	200ms	400ms	600ms	800ms	1.0s	1.2s	1.4s	1.6s	1.8s	2.0s	2.2s
▼ Scorekeep AWS::EC2::Instance															
Scorekeep	-	2.1 sec	✓												
SNS	400	728 ms	⚠	Subscribe											
▶ SNS AWS::SNS (Client Response)															

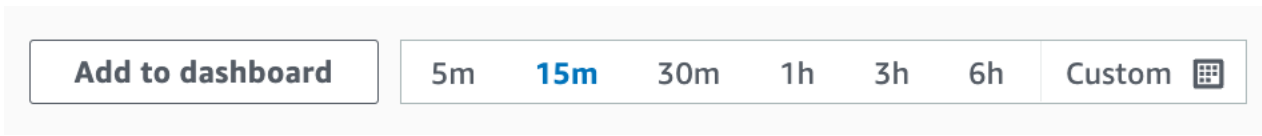
5.
- O X-Ray SDK captura automaticamente exceções lançadas pelos clientes instrumentados de SDK da AWS e registra o rastreamento da pilha.



CloudWatch console

Use o CloudWatch console

1. Abra a página do [mapa de rastreamento X-Ray](#) do CloudWatch console.
2. O console mostra uma representação do gráfico de serviço que o X-Ray gera com base nos dados de rastreamento enviados pelo aplicativo. Ajuste o período do mapa de serviço, se necessário, para garantir que ele exiba todos os rastreamentos desde que você iniciou o aplicativo web.



O mapa de rastreamento mostra o cliente do aplicativo web, a API em execução na Amazon EC2 e cada tabela do DynamoDB que o aplicativo usa. Cada solicitação para o aplicativo, até um número máximo configurável de solicitações por segundo, é rastreada quando atinge a API, gera solicitações para serviços subsequentes e é concluída.

Você pode escolher qualquer nó no gráfico do serviço para visualizar rastreamentos de solicitações que geraram tráfego nesse nó. No momento, o nó do Amazon SNS é laranja. Pesquise mais para descobrir o motivo.



Para encontrar a causa do erro

1. Escolha o nó chamado SNS. O painel de detalhes do nó SNS é exibido abaixo do mapa.
2. Escolha Visualizar rastreamentos para acessar a página Rastreamentos.
3. Adicione a parte inferior da página e selecione o rastreamento na lista Rastreamentos. Esse rastreamento não tem um método ou URL porque, em vez de ele ser registrado em resposta a uma solicitação recebida, seu registro foi feito durante o startup.

Traces Info

Find traces by typing a query, build a query using the Query refiners section, or [choose a sample query](#). You can also [find a trace by ID](#).

Filter by X-Ray group service(id(name: "SNS", type: "AWS::SNS"))

Run query 1 traces retrieved

Query refiners

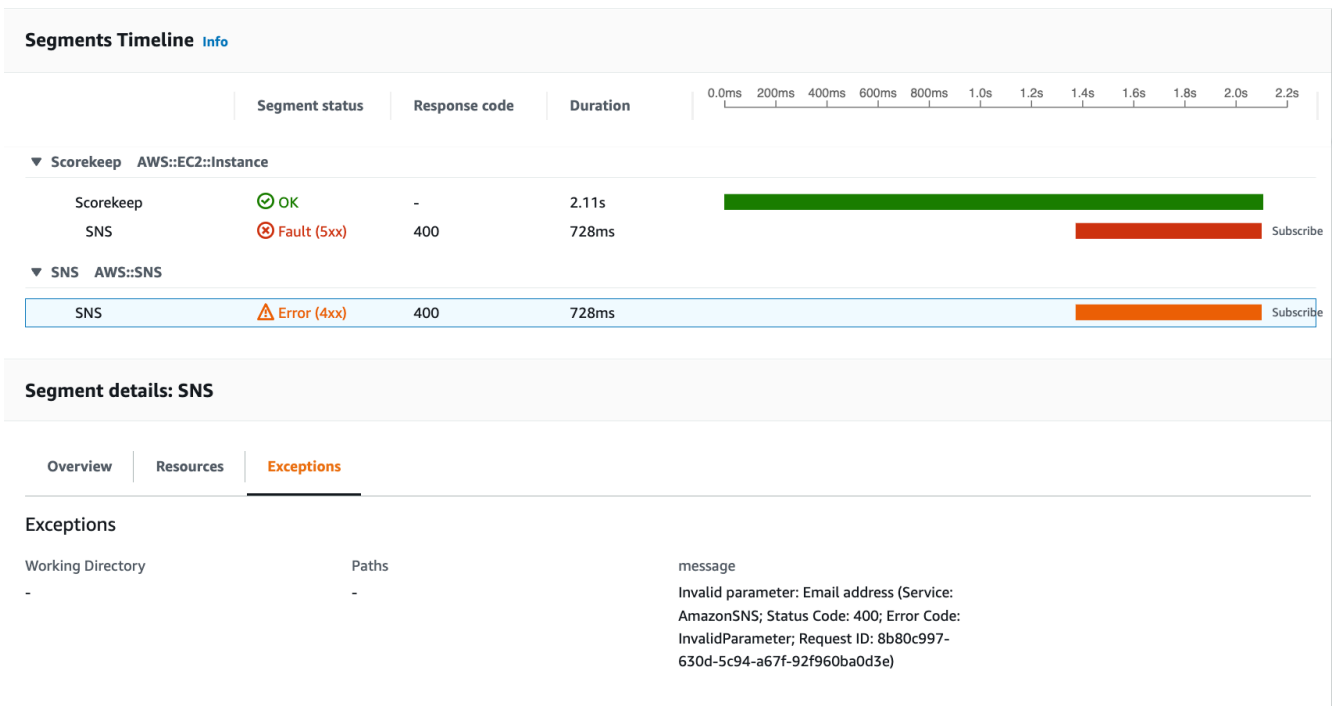
Traces (1) Add to dashboard

This table shows the most recent traces with an average response time of 2.11s. It shows as many as 1000 traces.

Start typing to filter trace list

ID	Trace status	Timestamp	Response code	Response Time	Duration
...86b347fc50bc57a992e9b835	OK	19.1min (2022-08-10 12:05:25)	-	2.11s	2.11s

4. Escolha o subsegmento do Amazon SNS na parte inferior da linha do tempo dos segmentos e selecione a guia Exceções do subsegmento do SNS para visualizar os detalhes da exceção.



A causa indica que o endereço de e-mail fornecido em uma chamada de `createSubscription` feita na classe `WebConfig` foi inválida. Na próxima seção, corrigiremos isso.

Configuração de notificações do Amazon SNS

O Scorekeep usa o Amazon SNS para enviar notificações quando os usuários concluem um jogo. Quando o aplicativo é iniciado, ele tenta criar uma assinatura para um endereço de e-mail definido em um parâmetro de CloudFormation pilha. Essa chamada está falhando no momento. Configure um e-mail de notificação para habilitar notificações e resolver as falhas destacadas no mapa de serviço.

Console de gerenciamento da AWS

Para configurar as notificações do Amazon SNS usando o Console de gerenciamento da AWS

1. Abra o [console do CloudFormation](#).
2. Escolha o botão de rádio ao lado do nome da pilha `scorekeep` na lista e selecione Atualizar.
3. A opção Usar modelo atual deve estar selecionada. Clique em Próximo na página Atualizar pilha.
4. Encontre o parâmetro E-mail na lista e substitua o valor padrão por um endereço de e-mail válido.

EcsInstanceTypeT3

Specifies the EC2 instance type for your container instances. Defaults to t3.micro.

Email**FrontendImageUri**

5. Role para o final da página e selecione Next (Próximo).
6. Role até a parte inferior da página de revisão, escolha a caixa de seleção reconhecendo que CloudFormation pode criar recursos do IAM com nomes personalizados e escolha Atualizar pilha.
7. A CloudFormation pilha agora está sendo atualizada. O status da pilha será UPDATE_IN_PROGRESS por cerca de cinco minutos antes de mudar para UPDATE_COMPLETE. O status será atualizado periodicamente ou você poderá atualizar a página.

AWS CLI

Para configurar as notificações do Amazon SNS usando o AWS CLI

1. Navegue até a pasta `xray-scorekeep/cloudformation/` que você criou anteriormente e abra o arquivo `cf-resources.yaml` em um editor de texto.
2. Encontre o Default valor no parâmetro Email e altere-o **UPDATE_ME** para um endereço de e-mail válido.

Parameters:**Email:**

Type: String

Default: UPDATE_ME # <- change to a valid abc@def.xyz email address

3. Na `cloudformation` pasta, atualize a CloudFormation pilha com o seguinte AWS CLI comando:

```
aws cloudformation update-stack --stack-name scorekeep --capabilities
"CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

4. Espere até que o status da CloudFormation pilha seja `UPDATE_COMPLETE`, o que levará alguns minutos. Use o AWS CLI comando a seguir para verificar o status:

```
aws cloudformation describe-stacks --stack-name scorekeep --query  
"Stacks[0].StackStatus"
```

Quando a atualização for concluída, o Scorekeep será reiniciado e criará uma assinatura para o tópico do SNS. Verifique seu e-mail e confirme a assinatura para ver as atualizações quando você concluir um jogo. Abra o mapa de serviço para verificar se as chamadas para o SNS não estão mais falhando.

Explorar o aplicativo de amostra

A aplicativo de exemplo é uma API da web HTTP em Java que é configurada para usar o X-Ray SDK para Java. Quando você implanta o aplicativo com o CloudFormation modelo, ele cria as tabelas do DynamoDB, o Amazon ECS Cluster e outros serviços necessários para executar o Scorekeep no ECS. Um arquivo de definição de tarefas para o ECS é criado por meio de CloudFormation. Esse arquivo define as imagens de contêiner usadas por tarefa em um cluster do ECS. Essas imagens são obtidas do ECR público oficial do X-Ray. A imagem de contêiner da API do Scorekeep tem a API compilada com o Gradle. A imagem de contêiner do front-end do Scorekeep atende ao front-end usando o servidor de proxy nginx. Esse servidor encaminha solicitações para caminhos que começam com `/api` à API.

Para instrumentar solicitações HTTP de entrada, o aplicativo adiciona o `TracingFilter` fornecido pelo SDK.

Example `src/main/java/scorekeep/WebConfig.java` - filtro de servlet

```
import javax.servlet.Filter;  
import com.amazonaws.xray.java.servlet.AWSXRayServletFilter;  
...  
  
@Configuration  
public class WebConfig {  
  
    @Bean  
    public Filter TracingFilter() {  
        return new AWSXRayServletFilter("Scorekeep");  
    }  
}
```

...

Esse filtro envia dados de rastreamento sobre todas as solicitações de entrada que o aplicativo atende, incluindo URL de solicitação, método, status de resposta, horário de início e de término.

A aplicativo também faz chamadas subsequentes para o DynamoDB usando o AWS SDK para Java. Para instrumentar essas chamadas, o aplicativo simplesmente usa os submódulos AWS relacionados ao SDK como dependências, e o X-Ray SDK for Java instrumenta automaticamente todos os clientes do SDK. AWS

A aplicativo usa o Docker para criar o código-fonte na instância com a Gradle Docker Image e um arquivo Scorekeep API Dockerfile para executar o JAR executável que o Gradle gera no respectivo ENTRYPOINT.

Example Exemplo do uso do Docker para criar imagem do Docker por meio do Gradle

```
docker run --rm -v /PATH/TO/SCOREKEEP_REPO/home/gradle/project -w /home/gradle/project
gradle:4.3 gradle build
```

Example Exemplo ENTRYPOINT do Dockerfile

```
ENTRYPOINT [ "sh", "-c", "java -Dserver.port=5000 -jar scorekeep-api-1.0.0.jar" ]
```

O arquivo build.gradle baixa os submódulos do SDK a partir do Maven durante a compilação declarando-os como dependências.

Example build.gradle -- dependências

```
...
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")
    }
}
```

```
}
}
```

Os submódulos core, AWS SDK e AWS SDK Instrumentor são tudo o que é necessário para instrumentar automaticamente qualquer chamada downstream feita com o SDK. AWS

Para retransmitir os dados do segmento bruto para a API do X-Ray, o daemon do X-Ray é necessário para escutar o tráfego na porta UDP 2000. Para fazer isso, o aplicativo executa o daemon do X-Ray em um contêiner que é implantado com o aplicativo Scorekeep no ECS como um contêiner auxiliar. Confira o tópico [Daemon do X-Ray](#) para obter mais informações.

Example Definição do contêiner do daemon do X-Ray em uma definição de tarefa do ECS

```
...
Resources:
  ScorekeepTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
        ...

        - Cpu: '256'
          Essential: true
          Image: amazon/aws-xray-daemon
          MemoryReservation: '128'
          Name: xray-daemon
          PortMappings:
            - ContainerPort: '2000'
              HostPort: '2000'
              Protocol: udp
          ...
```

O X-Ray SDK para Java oferece uma classe chamada `AWSXRay`, que fornece o registrador global, um `TracingHandler` que você pode usar para instrumentar seu código. Você pode configurar o gravador global para personalizar o `AWSXRayServletFilter` que cria segmentos para chamadas HTTP de entrada. A amostra inclui um bloco estático na classe `WebConfig` que configura o registrador global com plugins e regras de amostragem.

Example `src/main/java/scorekeep/WebConfig.java` - gravador

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
```

```

import com.amazonaws.xray.java.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());
        ...
    }
}

```

Este exemplo usa o builder para carregar regras de amostragem a partir de um arquivo chamado `sampling-rules.json`. As regras de amostragem determinam a taxa na qual o SDK registra segmentos para as solicitações recebidas.

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
}
}

```

O arquivo de regras de amostragem define quatro regras de amostragem personalizadas e a regra padrão. Para cada solicitação de entrada, o SDK avalia as regras personalizadas na ordem em que são definidas. O SDK aplica a primeira regra que corresponde ao método, ao caminho e ao nome do serviço da solicitação. Para o Scorekeep, a primeira regra detecta todas as solicitações POST (chamadas de criação de recursos) aplicando um destino fixo de uma solicitação por segundo e uma taxa de 1.0 ou 100% de solicitações após o destino fixo ser alcançado.

As outras três regras personalizadas se aplicam a uma taxa fixa de 5%, sem destino fixo para leituras de estado, jogo e sessão (solicitações GET). Isso minimiza o número de rastreamentos das chamadas periódicas que o front-end faz automaticamente a cada poucos segundos para garantir que o conteúdo esteja atualizado. Para todas as outras solicitações, o arquivo define uma taxa padrão de uma solicitação por segundo e uma taxa de 10%.

O aplicativo de amostra também mostra como usar recursos avançados, tais como instrumentação de cliente do SDK manual, criação de subsegmentos adicionais e chamadas HTTP de saída. Para obter mais informações, consulte [AWS X-Ray aplicação de amostra](#).

Opcional: política de menor privilégio

Os contêineres do ECS do Scorekeep acessam recursos usando políticas de acesso total, como `AmazonSNSFullAccess` e `AmazonDynamoDBFullAccess`. Usar políticas de acesso total não é a prática recomendada para aplicativos de produção. O exemplo a seguir atualiza a política do IAM do DynamoDB para melhorar a segurança do aplicativo. Para saber mais sobre as melhores práticas de segurança nas políticas do IAM, consulte [Gerenciamento de identidade e acesso para AWS X-Ray](#).

Exemplo modelo `cf-resources.yaml` Definição de função ECSTask

```
ECSTaskRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        -
          Effect: "Allow"
          Principal:
            Service:
              - "ecs-tasks.amazonaws.com"
          Action:
            - "sts:AssumeRole"
    ManagedPolicyArns:
      - "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
      - "arn:aws:iam::aws:policy/AmazonSNSFullAccess"
      - "arn:aws:iam::aws:policy/AWSXrayFullAccess"
    RoleName: "scorekeepRole"
```

Para atualizar sua política, primeiro identifique os ARNs dos recursos do DynamoDB. Em seguida, use o ARN em uma política personalizada do IAM. Por fim, aplique essas políticas ao perfil de instância.

Para identificar o ARN do seu recurso do DynamoDB:

1. Abra o [console do DynamoDB](#).
2. Selecione Tabelas na barra de navegação à esquerda.

3. Escolha qualquer uma das opções `scorekeep-*` para exibir a página de detalhes da tabela.
4. Na guia Visão geral, escolha Informações adicionais para expandir a seção e visualizar o nome do recurso da Amazon (ARN). Copie o valor.
5. Insira o ARN na política do IAM a seguir, substituindo os valores `AWS_REGION` e `AWS_ACCOUNT_ID` por sua região específica e ID da conta. Essa nova política permite somente as ações especificadas, em vez da política `AmazonDynamoDBFullAccess`, que permite qualquer ação.

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-*"
    }
  ]
}
```

As tabelas criadas pelo aplicativo seguem uma convenção de nomenclatura consistente. Você pode usar o formato `scorekeep-*` para indicar todas as tabelas do Scorekeep.

Alterar a política do IAM

1. Abra o [Perfil da tarefa do Scorekeep \(scorekeepRole\)](#) no console do IAM.

2. Marque a caixa de seleção ao lado da política `AmazonDynamoDBFullAccess` e selecione **Remover** para remover essa política.
3. Escolha **Adicionar permissões**, depois **Anexar políticas** e, finalmente, **Criar política**.
4. Escolha a guia **JSON** e cole a política criada acima.
5. Na parte inferior da página, selecione **Próximo: Tags**.
6. Escolha **Próximo: Revisar** na parte inferior da página.
7. Para **Nome**, atribua um nome para a política.
8. Escolha **Criar política** na parte inferior da página.
9. Anexe a política recém-criada ao perfil `scorekeepRole`. Pode levar alguns minutos para que a política anexada entre em vigor.

Se você anexou a nova política à `scorekeepRole` função, deverá desanexá-la antes de excluir a CloudFormation pilha, pois essa política anexada impedirá que a pilha seja excluída. A política pode ser desanexada automaticamente excluindo a política.

Remover sua política do IAM personalizada

1. Abra o [console do IAM](#).
2. Escolha **Políticas** na barra de navegação à esquerda.
3. Pesquise o nome da política personalizada que você criou anteriormente nesta seção e escolha o botão de rádio ao lado do nome da política para destacá-la.
4. Escolha o menu suspenso **Ações** e selecione **Excluir**.
5. Digite o nome da política personalizada e escolha **Excluir** para confirmar a exclusão. Isso separará automaticamente a política do perfil `scorekeepRole`.

Limpeza

Siga estas etapas para excluir os recursos do aplicativo Scorekeep:

Note

Se você criou e anexou políticas personalizadas usando a seção anterior deste tutorial, deverá remover a política do `scorekeepRole` antes de excluir a CloudFormation pilha.

Console de gerenciamento da AWS

Exclua o aplicativo de amostra usando o Console de gerenciamento da AWS

1. Abra o [console do CloudFormation](#).
2. Escolha o botão de opção ao lado do nome da pilha scorekeep na lista e selecione Excluir.
3. A CloudFormation pilha agora está sendo excluída. O status da pilha permanecerá DELETE_IN_PROGRESS por alguns minutos até que todos os recursos sejam excluídos. O status será atualizado periodicamente ou você poderá atualizar a página.

AWS CLI

Exclua o aplicativo de amostra usando o AWS CLI

1. Digite o seguinte AWS CLI comando para excluir a CloudFormation pilha:

```
aws cloudformation delete-stack --stack-name scorekeep
```

2. Espere até que a CloudFormation pilha não exista mais, o que levará cerca de cinco minutos. Use o AWS CLI comando a seguir para verificar o status:

```
aws cloudformation describe-stacks --stack-name scorekeep --query  
"Stacks[0].StackStatus"
```

Próximas etapas

Saiba mais sobre o X-Ray no próximo capítulo, [AWS X-Ray Conceitos do](#).

Para instrumentar seu próprio aplicativo, saiba mais sobre o X-Ray SDK for Java ou um dos outros X-Ray SDKs:

- X-Ray SDK para Java: [AWS X-Ray SDK for Java](#)
- X-Ray SDK para Node.js: [AWS X-Ray SDK para Node.js](#)
- X-Ray SDK para .NET: [AWS X-Ray SDK para .NET](#)

Para executar o daemon X-Ray localmente ou ligado AWS, consulte. [AWS X-Ray daemon](#)

Para contribuir com o aplicativo de amostra em GitHub, consulte [eb-java-scorekeep](#).

Instrumentando manualmente os clientes do AWS SDK

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O X-Ray SDK for Java instrumenta automaticamente AWS todos os clientes SDK quando [você inclui AWS o submódulo SDK Instrumentor](#) em suas dependências de compilação.

Você pode desativar a instrumentação automática do cliente removendo o submódulo Instrumentor. Isso permite que você instrumente alguns clientes manualmente enquanto ignora outros, ou use manipuladores de rastreamento diferentes para clientes diferentes.

Para ilustrar o suporte à instrumentação de clientes AWS SDK específicos, o aplicativo passa um manipulador de rastreamento para `AmazonDynamoDBClientBuilder` um manipulador de solicitações no modelo de usuário, jogo e sessão. Essa alteração de código instrui o SDK a instrumentar todas as chamadas para o DynamoDB usando esses clientes.

Exemplo [src/main/java/scorekeep/SessionModel.java](#): instrumentação manual de cliente de SDK da AWS

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.handlers.TracingHandler;

public class SessionModel {
    private AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
        .withRegion(Constants.REGION)
        .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
        .build();
    private DynamoDBMapper mapper = new DynamoDBMapper(client);
```

Se você remover o submódulo AWS SDK Instrumentor das dependências do projeto, somente os clientes AWS SDK instrumentados manualmente aparecerão no mapa de rastreamento.

Criar subsegmentos adicionais

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Na classe de modelo de usuário, o aplicativo cria manualmente subsegmentos para agrupar todas as chamadas de downstream feitas na função `saveUser` e adiciona metadados.

Exemplo [src/main/java/scorekeep/UserModel.java](#) – Subsegmentos personalizados

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Subsegment;
...
public void saveUser(User user) {
    // Wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## UserModel.saveUser");
    try {
        mapper.save(user);
    } catch (Exception e) {
        subsegment.addException(e);
        throw e;
    } finally {
        AWSXRay.endSubsegment();
    }
}
```

Gravando anotações, metadados e usuário IDs

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Na classe de modelos de jogos, o aplicativo registra objetos Game em um bloco de [metadados](#) toda vez que salva um jogo no DynamoDB. Separadamente, o aplicativo grava o jogo IDs em [anotações](#) para uso com expressões de [filtro](#).

Exemplo [src/main/java/scorekeep/GameModel.java](#): anotações e metadados

```
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();
    }
}
```

```
}  
}
```

No controlador de movimentação, o aplicativo registra o [usuário IDs](#) com `setUser`. IDsOs usuários são registrados em um campo separado nos segmentos e são indexados para uso com a pesquisa.

Example [src/main/java/scorekeep/MoveController.java](#) — ID do usuário

```
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);  
}
```

Instrumentar chamadas HTTP de saída

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

A classe de usuário de fábrica mostra como o aplicativo usa a versão `HTTPClientBuilder` do X-Ray SDK para Java para instrumentar chamadas HTTP de saída.

Example [src/main/java/scorekeep/UserFactory.java](#) — `HTTPClient` instrumentação

```
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();  
    }  
}
```

Se você usa atualmente `org.apache.http.impl.client.HttpClientBuilder`, pode simplesmente trocar a declaração de importação dessa classe por uma para `com.amazonaws.xray.proxies.apache.http.HttpClientBuilder`.

Instrumentação de chamadas para um banco de dados PostgreSQL

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O arquivo `application-pgsql.properties` adiciona o interceptor de rastreamento do X-Ray PostgreSQL à fonte de dados criada em [RdsWebConfig.java](#).

Exemplo [application-pgsql.properties](#): instrumentação de banco de dados PostgreSQL

```
spring.datasource.continue-on-error=true
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.jdbc-interceptors=com.amazonaws.xray.sql.postgres.TracingInterceptor
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

 Note

Consulte [Configuração de bancos de dados com Elastic Beanstalk](#) no AWS Elastic Beanstalk Guia do desenvolvedor para detalhes sobre como adicionar um banco de dados PostgreSQL ao ambiente de aplicativos.

A página de demonstração do X-Ray na ramificação `xray` inclui uma demonstração que usa a fonte de dados instrumentada para gerar rastreamentos que mostram informações sobre as consultas SQL que ela gera. Navegue até o caminho `/#/xray` no aplicativo em execução ou selecione `Powered by AWS X-Ray`(Desenvolvido pelo `&xraylong`;) na barra de navegação, para ver a página de demonstração.

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.

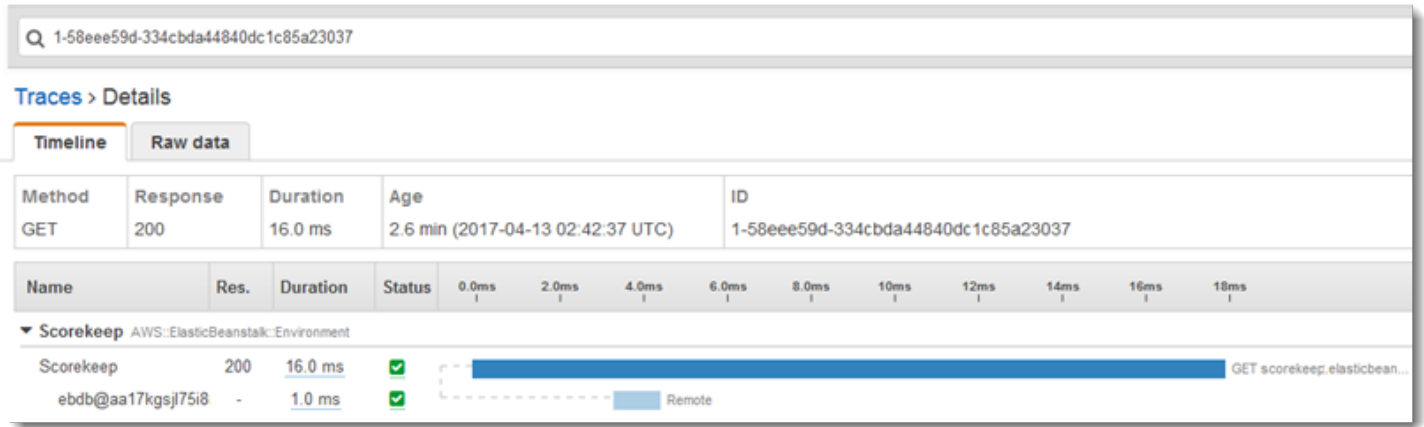
Trace SQL queries

View traces in AWS X-Ray

ID	Winner	Loser
1	Mugur	Gheorghită
2	Paula	Adorján
3	Αρχίας	Stela
4	付	Pərvanə

Selecione Trace SQL queries para simular sessões de jogos e armazenar os resultados no banco de dados anexado. Em seguida, escolha Exibir traços no AWS X-Ray para ver uma lista filtrada dos rastreamentos que atingem a `/api/history` rota da API.

Selecione um dos rastreamentos na lista para ver o cronograma, incluindo a consulta SQL.



Funções de instrumentação AWS Lambda

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O Scorekeep usa duas AWS Lambda funções. A primeira é a função Node.js na ramificação `lambda` que gera nomes aleatórios para novos usuários. Quando um usuário cria uma sessão sem informar um nome, o aplicativo chama uma função denominada `random-name` com o AWS SDK para Java. O X-Ray SDK for Java registra informações sobre a chamada para o Lambda em um subsegmento, como qualquer outra chamada feita com um cliente SDK instrumentado. AWS

 Note

A execução da função `random-name` do Lambda requer a criação de recursos adicionais fora do ambiente do Elastic Beanstalk. Consulte o arquivo `leia-me` para obter mais informações e instruções: [AWS Lambda Integration](#).

A segunda função, `scorekeep-worker`, é uma função Python que pode ser executada independentemente da API do Scorekeep. Quando um jogo termina, a API grava o ID da sessão e o ID do jogo em uma fila do SQS. A função de operador lê itens da fila e chama a API do Scorekeep para criar registros completos de cada sessão do jogo para armazenamento no Amazon S3.

O Scorekeep inclui CloudFormation modelos e scripts para criar as duas funções. Como você precisa empacotar o X-Ray SDK com o código da função, os modelos criam as funções sem nenhum código. Quando você implanta o Scorekeep, um arquivo de configuração incluído na pasta `.ebextensions` cria um pacote de origem que inclui o SDK e atualiza o código da função e a configuração com a AWS Command Line Interface.

Funções

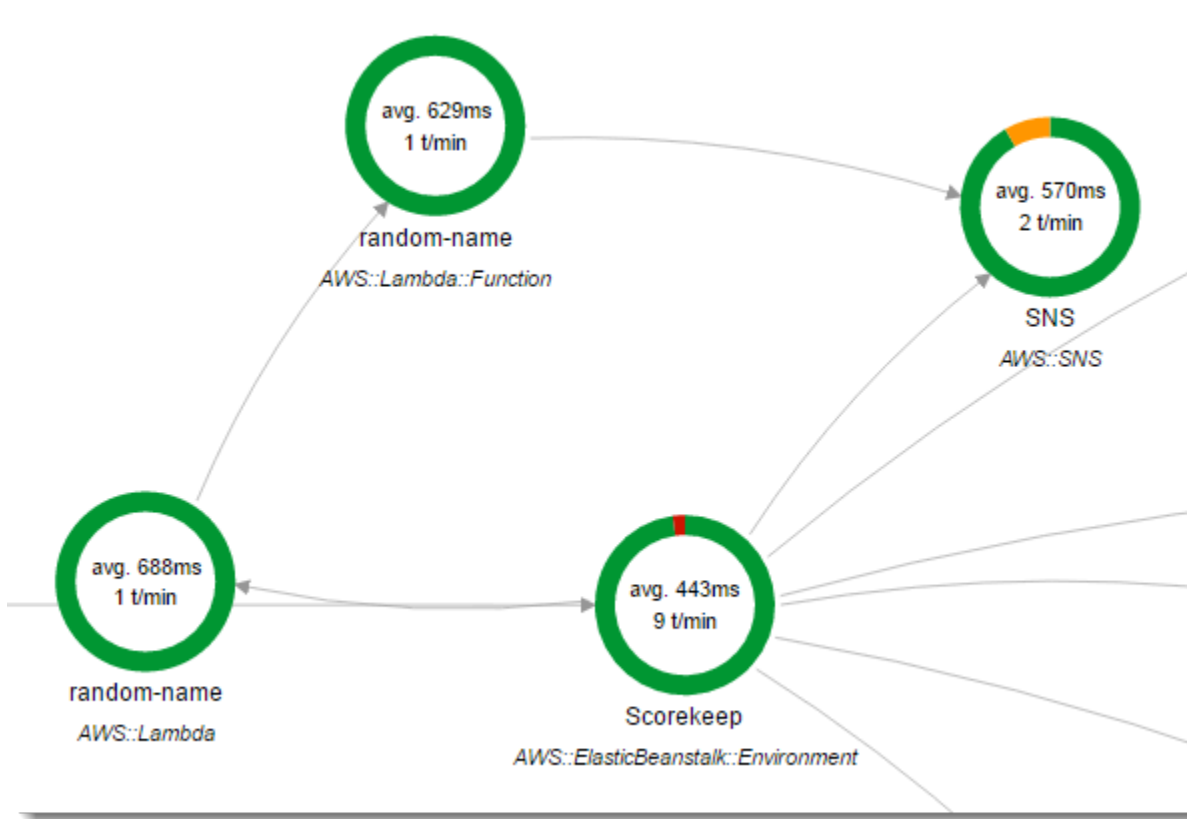
- [Nome aleatório](#)
- [Operador](#)

Nome aleatório

O Scorekeep chama a função de nome aleatório quando um usuário inicia uma sessão do jogo sem fazer login ou especificando um nome de usuário. Quando o Lambda processa a chamada a `random-name`, ele lê o [cabeçalho de rastreamento](#), que contém o ID de rastreamento e a decisão de amostragem gravados pelo X-Ray SDK para Java.

Para cada solicitação amostrada, o Lambda executa o daemon do X-Ray e grava dois segmentos. O primeiro registra informações sobre a chamada ao Lambda que invoca a função. Esse segmento contém as mesmas informações que o subsegmento registrado pelo Scorekeep, mas do ponto de vista do Lambda. O segundo segmento representa o trabalho que a função faz.

O Lambda passa o segmento de função para o X-Ray SDK por meio do contexto da função. Ao instrumentar uma função do Lambda, você não usa o SDK para [criar um segmento para as solicitações de entrada](#). O Lambda fornece o segmento e você usa o SDK para instrumentar clientes e gravar subsegmentos.



A função `random-name` é implementada em Node.js. Ele usa o SDK do Node.js para JavaScript enviar notificações com o Amazon SNS e o X-Ray SDK para Node.js para AWS instrumentar o cliente do SDK. Para gravar anotações, a função cria um subsegmento personalizado com `AWSXRay.captureFunc` e grava anotações na função instrumentada. No Lambda, não é possível gravar anotações diretamente no segmento da função, apenas em um subsegmento que você criar.

Exemplo [function/index.js](#): função do lambda de nome aleatório

```
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 random 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;
```

Essa função é criada automaticamente quando você implanta o aplicativo de amostra para o Elastic Beanstalk. A ramificação `xray` inclui um script para criar uma função do Lambda vazia. Os arquivos de configuração na `.ebextensions` pasta criam o pacote de funções npm `install` durante a implantação e, em seguida, atualizam a função Lambda com a CLI AWS.

Operador

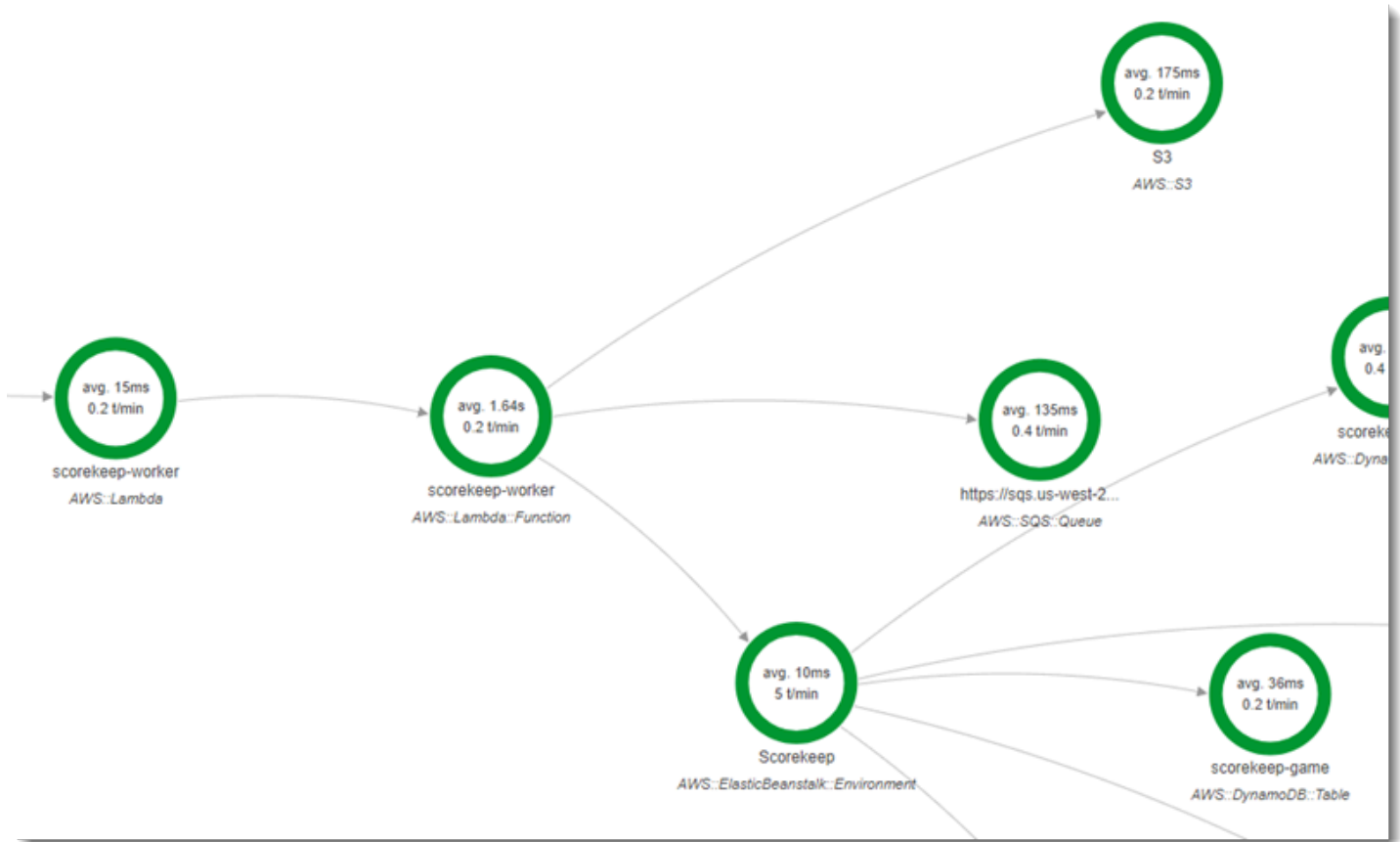
A função de operador instrumentada é fornecida em sua própria ramificação `xray-worker`, uma vez que ela não pode executar a menos que você crie a função de operador e os recursos relacionados antes. Consulte o [leia-me da ramificação](#) para instruções.

A função é acionada por um evento agrupado da Amazon CloudWatch Events a cada 5 minutos. Quando ela é executada, a função obtém um item de uma fila do Amazon SQS que o Scorekeep gerencia. Cada mensagem contém informações sobre um jogo concluído.

O operador obtém o registro de jogos e documentos de outras tabelas às quais o registro do jogo faz referência. Por exemplo, o registro do jogo no DynamoDB inclui uma lista de movimentos executados

durante o jogo. A lista não contém os movimentos em si, mas sim IDs os movimentos armazenados em uma tabela separada.

As sessões e os estados também são armazenados como referências. Isso impede que as entradas na tabela de jogos sejam muito grandes, mas requer chamadas adicionais para obter todas as informações sobre o jogo. O operador cancela as referências a todas essas entradas e cria um registro completo do jogo como um único documento no Amazon S3. Quando quiser analisar os dados, você poderá executar consultas diretamente no Amazon S3 com o Amazon Athena sem executar migrações de dados que exigem muita leitura para obter os dados do DynamoDB.



A função de operador tem o rastreamento ativo em sua configuração no AWS Lambda. Ao contrário da função de nome aleatório, o trabalhador não recebe uma solicitação de um aplicativo instrumentado e, portanto, AWS Lambda não recebe um cabeçalho de rastreamento. Com o rastreamento ativo, o Lambda cria o ID de rastreamento e toma as decisões de amostragem.

O X-Ray SDK para Python está apenas algumas linhas na parte superior da função que importa o SDK e executa `patch_all` sua função para corrigir o e que ele usa para AWS SDK for Python (Boto) chamar o Amazon SQS HTTPclients e o Amazon S3. Quando o operador chama a API do

Scorekeep, o SDK adiciona o [cabeçalho de rastreamento](#) à solicitação para rastrear chamadas por meio da API.

Example [_lambda/scorekeep-worker/scorekeep-worker.py](#) -- função do lambda do operador

```
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
    )
    ...
```

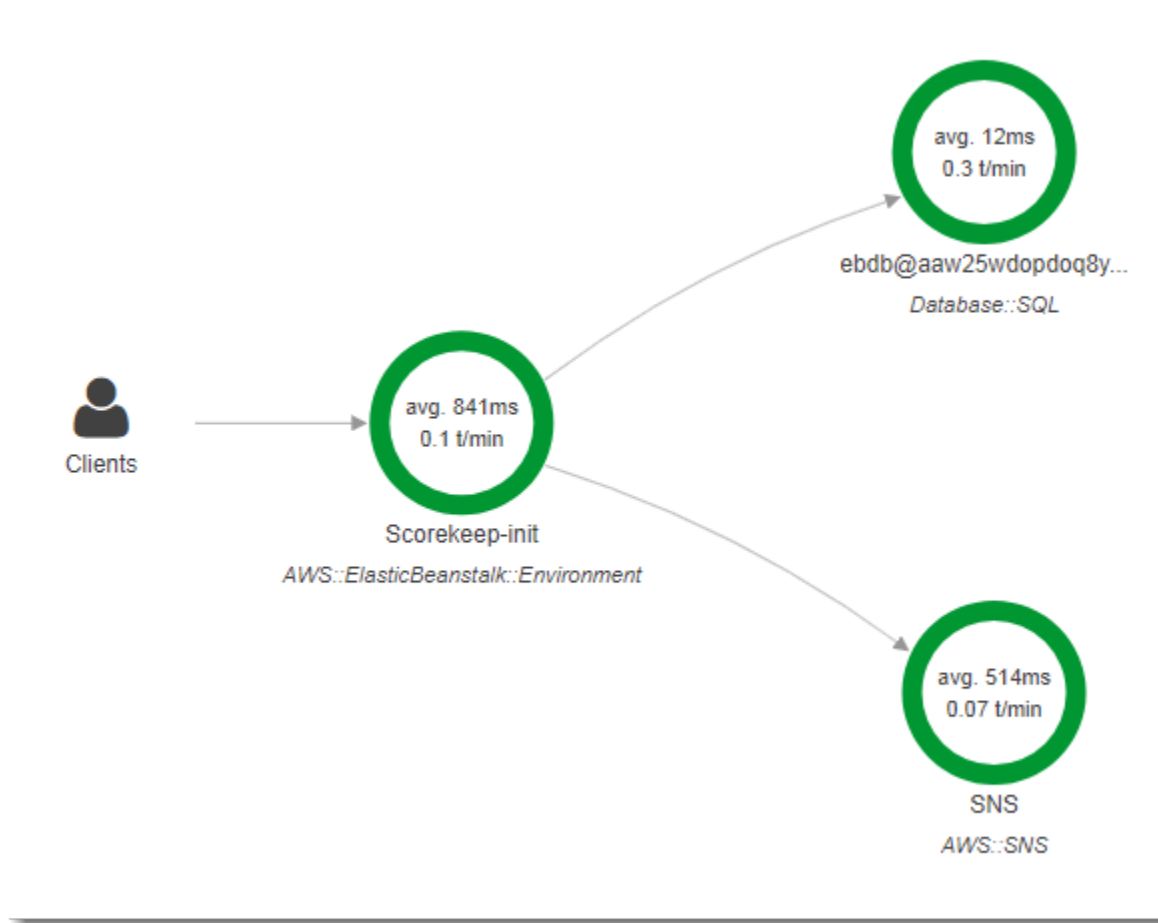
Instrumentar código de inicialização

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O X-Ray SDK para Java cria segmentos para solicitações de entrada automaticamente. Desde que uma solicitação esteja dentro do escopo, você pode usar clientes instrumentados e gravar subsegmentos sem problema. No entanto, se você tentar usar um cliente instrumentado no código de inicialização, obterá um [SegmentNotFoundException](#).

O código de inicialização é executado fora do request/response fluxo padrão de um aplicativo web, então você precisa criar segmentos manualmente para instrumentá-lo. O Scorekeep mostra a instrumentação do código de inicialização em seus arquivos WebConfig. O Scorekeep chama um banco de dados SQL e o Amazon SNS durante a inicialização.



A classe `WebConfig` padrão cria uma assinatura do Amazon SNS para notificações. Para fornecer um segmento para o X-Ray SDK gravar quando o cliente do Amazon SNS for usado, o `Scorekeep` chama `beginSegment` e `endSegment` no gravador global.

Exemplo [src/main/java/scorekeep/WebConfig.java](#): cliente de SDK da AWS instrumentado no código de inicialização

```

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();

```

Em `RdsWebConfig`, que é usada pelo Scorekeep quando um banco de dados do Amazon RDS está conectado, a configuração também cria um segmento para o cliente SQL que o Hibernate usa ao aplicar o esquema do banco de dados durante a inicialização.

Exemplo [src/main/java/scorekeep/RdsWebConfig.java](#): cliente de banco de dados SQL instrumentado no código de inicialização

```
@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` é executado automaticamente e usa um cliente SQL. Uma vez que o cliente é instrumentado, o Scorekeep deve substituir a implementação padrão e fornecer um segmento para o SDK usar quando o cliente for invocado.

Scripts de instrumentação

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray](#)

[SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você também pode instrumentar código que não faz parte de seu aplicativo. Quando o daemon do X-Ray está em execução, ele retransmite todos os segmentos que recebe para o X-Ray, mesmo que eles não tenham sido gerados pelo X-Ray SDK. O Scorekeep usa seus próprios scripts para instrumentar o build que compila o aplicativo durante a implantação.

Example [bin/build.sh](#): script de compilação instrumentado

```
SEGMENT=$(python bin/xray_start.py)
gradle build --quiet --stacktrace &> /var/log/gradle.log; GRADLE_RETURN=$?
if (( GRADLE_RETURN != 0 )); then
    echo "Gradle failed with exit status $GRADLE_RETURN" >&2
    python bin/xray_error.py "$SEGMENT" "$(cat /var/log/gradle.log)"
    exit 1
fi
python bin/xray_success.py "$SEGMENT"
```

[xray_start.py](#), [xray_error.py](#) and [xray_success.py](#) são scripts simples do Python que constroem objetos de segmento, convertem-nos em documentos JSON e envia-os para o daemon por meio de UDP. Se a compilação do Gradle falhar, você poderá encontrar a mensagem de erro clicando no nó scorekeep-build no mapa de serviço do console do X-Ray.



Traces > Details

Timeline

Raw data

Method	Response	Duration	Age	ID
--	--	14.6 sec	4.5 min (2017-09-14 01:25:01 UTC)	1-59b9da6d-ab8ca2666217b31a03eff86d

Name	Res.	Duration	Status	0.0ms	2.0s	4.0s	6.0s	8.0s	10s	12s	14s	16s
▼ Scorekeep-build												
Scorekeep-build	-	14.6 sec		--								

Segment - Scorekeep-build



Overview

Resources

Annotations

Metadata

Exceptions

Working directory /var/app/current
 Paths /var/app/current/src/main/java/scorekeep/

Cause

```
/var/app/staging/src/main/java/scorekeep/RdsWebConfig.java:89: error: cannot find symbol
    AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new EC2Plugin()).withPlugin(new ElasticBeanstalkPlugin());
                                                                    ^
```

symbol: class ElasticBeanstalkPlugin

location: class RdsWebConfig

1 error

FAILURE: Build failed with an exception.

Close

Instrumentar o cliente do aplicativo web

 Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais

informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Na ramificação [xray-cognito](#), o Scorekeep usa o Amazon Cognito para permitir que os usuários criem uma conta e façam login com ela para recuperar informações de um grupo de usuários do Amazon Cognito. Quando um usuário faz login, o Scorekeep usa um pool de identidade do Amazon Cognito para obter credenciais AWS temporárias para uso com o AWS SDK para JavaScript

O grupo de identidades é configurado para permitir que os usuários conectados rastreiem dados para o AWS X-Ray. O aplicativo web usa essas credenciais para gravar o ID do usuário conectado, o caminho do navegador e a visualização de chamadas do cliente para a API do Scorekeep.

A maior parte do trabalho é feita em uma classe de serviço chamada `xray`. Essa classe de serviço oferece métodos para gerar os identificadores necessários, criar segmentos em andamento, finalizar segmentos e enviar documentos de segmentos à API do X-Ray.

Exemplo [public/xray.js](#): gravar e carregar segmentos

```
...
service.beginSegment = function() {
  var segment = {};
  var traceId = '1-' + service.getHexTime() + '-' + service.getHexId(24);

  var id = service.getHexId(16);
  var startTime = service.getEpochTime();

  segment.trace_id = traceId;
  segment.id = id;
  segment.start_time = startTime;
  segment.name = 'Scorekeep-client';
  segment.in_progress = true;
  segment.user = sessionStorage['userid'];
  segment.http = {
    request: {
      url: window.location.href
    }
  };

  var documents = [];
  documents[0] = JSON.stringify(segment);
  service.putDocuments(documents);
}
```

```

    return segment;
}

service.endSegment = function(segment) {
    var endTime = service.getEpochTime();
    segment.end_time = endTime;
    segment.in_progress = false;
    var documents = [];
    documents[0] = JSON.stringify(segment);
    service.putDocuments(documents);
}

service.putDocuments = function(documents) {
    var xray = new AWS.XRay();
    var params = {
        TraceSegmentDocuments: documents
    };
    xray.putTraceSegments(params, function(err, data) {
        if (err) {
            console.log(err, err.stack);
        } else {
            console.log(data);
        }
    })
}

```

Esses métodos são chamados no cabeçalho e nas funções `transformResponse` nos serviços do recurso que o aplicativo web usa para chamar a API do Scorekeep. Para incluir o segmento do cliente no mesmo rastreamento que o segmento gerado pela API, a aplicação web deve incluir o ID do rastreamento e o ID do segmento em um [cabeçalho de rastreamento](#) (X-Amzn-Trace-Id) que o X-Ray SDK possa ler. Quando a aplicação Java instrumentada recebe uma solicitação com esse cabeçalho, o X-Ray SDK para Java usa o mesmo ID de rastreamento e torna o segmento do cliente da aplicação web o pai do respectivo segmento.

Exemplo [public/app/services.js](#): gravar segmentos para chamadas de recurso angular e gravar cabeçalhos de rastreamento

```

var module = angular.module('scorekeep');
module.factory('SessionService', function($resource, api, XRay) {
    return $resource(api + 'session/:id', { id: '@_id' }, {
        segment: {},
        get: {

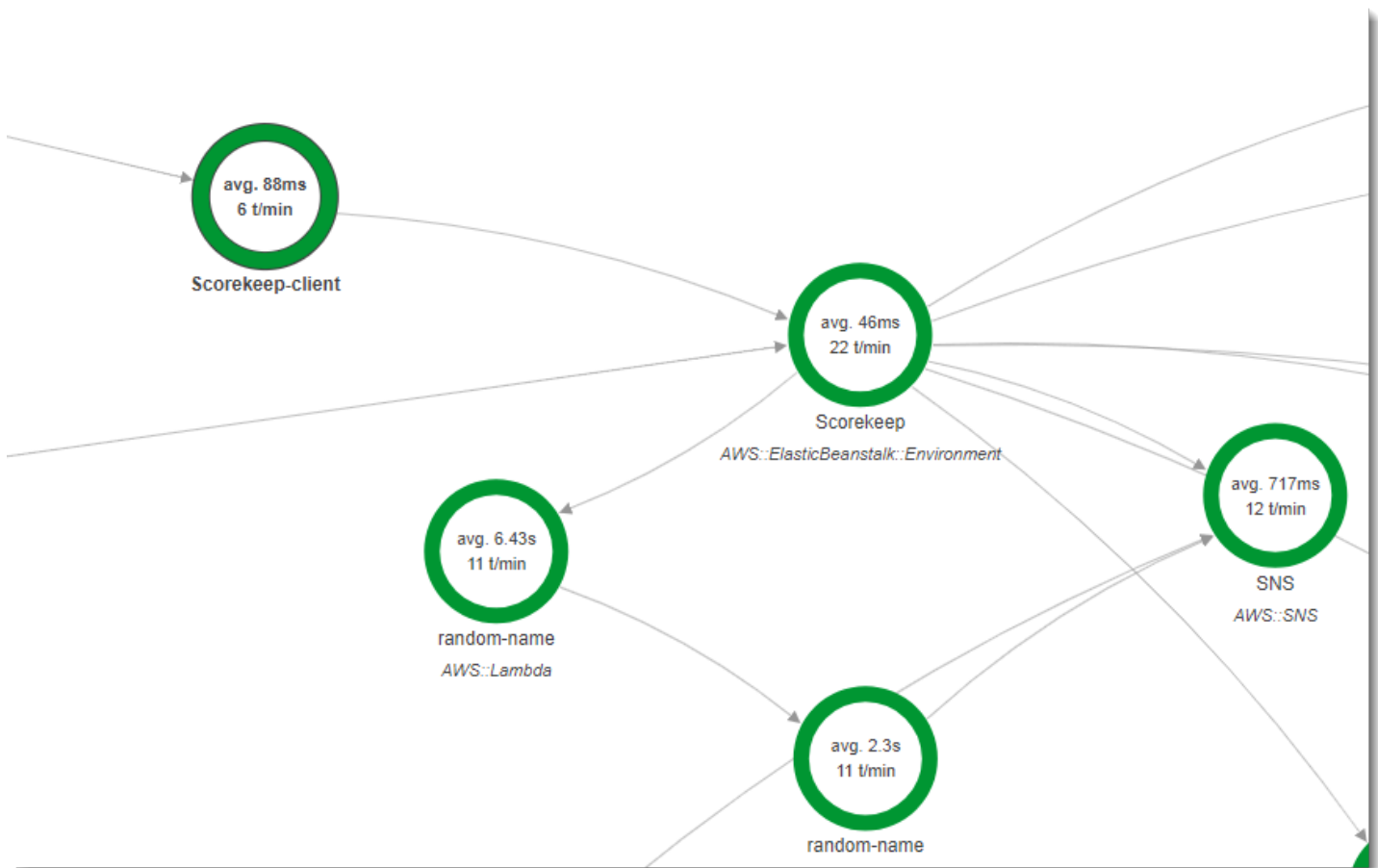
```

```

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);
},
},
...

```

O mapa do serviço resultante inclui um nó para o cliente do aplicativo web.



Os rastreamentos que incluem segmentos do aplicativo web mostram a URL que o usuário vê no navegador (caminhos começando com `/#/`). Sem instrumentação de cliente, você só obtém a URL do recurso da API que o aplicativo web chama (caminhos começando com `/api/`).

Trace overview

Group by: URL ▼

URL ▼	Avg response time ▼
http://scorekeep.elasticbeanstalk.com/#/	86.2 ms
http://scorekeep.elasticbeanstalk.com/#/session/4ORP7OB5/47H4SETD	58.5 ms
http://scorekeep.elasticbeanstalk.com/#/game/4ORP7OB5/A94SAFFD/47H4SETD	255 ms

Usar clientes instrumentais em threads de operador

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O Scorekeep usa um thread de operador para publicar uma notificação para o Amazon SNS quando um usuário ganha um jogo. A publicação da notificação demora mais que o restante das operações de solicitação combinadas, e não afeta o cliente ou o usuário. Portanto, a execução da tarefa de forma assíncrona é uma boa maneira de melhorar o tempo de resposta.

No entanto, o X-Ray SDK para Java não sabe qual segmento estava ativo quando o thread foi criado. Como resultado, quando você tenta usar o AWS SDK para Java cliente instrumentado dentro do encadeamento, ele lança um `SegmentNotFoundException`, travando o encadeamento.

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.java:
        ...

```

Para corrigir isso, a aplicação usa `GetTraceEntity` para obter uma referência ao segmento no thread principal e `Entity.run()` para passar o segmento de volta para o gravador no thread de operador.

Exemplo [src/main/java/scorekeep/MoveFactory.java](#): passar contexto de rastreamento para um thread de operador

```

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();
        }
    }
}

```

Como a solicitação agora é resolvida antes da chamada para o Amazon SNS, a aplicação cria um subsegmento separado para o thread. Isso evita que o X-Ray SDK feche o segmento antes de registrar a resposta do Amazon SNS. Se nenhum subsegmento estava aberto quando o Scorekeep resolveu a solicitação, a resposta do Amazon SNS poderá ser perdida.



Consulte [Passar o contexto do segmento entre threads em um aplicativo multithreaded](#) para obter mais informações sobre multithreading.

AWS X-Ray daemon

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Note

Agora você pode usar o CloudWatch agente para coletar métricas, registros e rastreamentos de EC2 instâncias da Amazon e servidores locais. CloudWatch O agente versão 1.300025.0 e posterior pode coletar traços do nosso cliente [OpenTelemetry](#) X-Ray e enviá-los para o [SDKsX-Ray](#). Usar o CloudWatch agente em vez do AWS Distro for OpenTelemetry (ADOT) Collector ou do daemon X-Ray para coletar traços pode ajudar a reduzir o número de agentes que você gerencia. Consulte o tópico do [CloudWatch agente](#) no Guia do CloudWatch usuário para obter mais informações.

O AWS X-Ray daemon é um aplicativo de software que escuta o tráfego na porta UDP 2000, reúne dados brutos do segmento e os retransmite para a API. AWS X-Ray O daemon funciona em conjunto com o AWS X-Ray SDKs e deve estar em execução para que os dados enviados pelo SDKs possam chegar ao serviço X-Ray. O daemon do X-Ray é um projeto de código-fonte aberto. Você pode acompanhar o projeto e enviar problemas e pull requests em GitHub: github.com/aws/aws-xray-daemon

Em seguida AWS Lambda AWS Elastic Beanstalk, use a integração desses serviços com o X-Ray para executar o daemon. O Lambda executa o daemon automaticamente sempre que uma função é invocada para uma solicitação amostrada. No Elastic Beanstalk, [use a opção de configuração XRayEnabled](#) para executar o daemon nas instâncias do ambiente. Para obter mais informações, consulte .

Para executar o daemon X-Ray localmente, localmente ou em outro local, baixe-o Serviços da AWS, [execute-o](#) e, em seguida, [conceda permissão para carregar documentos](#) do segmento no X-Ray.

Download do daemon

Você pode baixar o daemon do Amazon S3, Amazon ECR ou Docker Hub e, em seguida, executá-lo localmente ou instalá-lo em uma EC2 instância da Amazon na inicialização.

Amazon S3

Instaladores e executáveis do daemon do X-Ray

- Linux (executável): [aws-xray-daemon-linux-3.x.zip](#) ([sig](#))
- Linux (instalador RPM): [aws-xray-daemon-3.x.rpm](#)
- Linux (instalador DEB): [aws-xray-daemon-3.x.deb](#)
- Linux (ARM64, executável) — [aws-xray-daemon-linux-arm64-3.x.zip](#)([sig](#))
- Linux (ARM64, instalador RPM) — [aws-xray-daemon-arm64-3.x.rpm](#)
- Linux (ARM64, instalador de DEB) — [aws-xray-daemon-arm64-3.x.deb](#)
- OS X (executável): [aws-xray-daemon-macos-3.x.zip](#) ([sig](#))
- Windows (executável): [aws-xray-daemon-windows-process-3.x.zip](#) ([sig](#))
- Windows (serviço): [aws-xray-daemon-windows-service-3.x.zip](#) ([sig](#))

Esses links sempre apontam para a versão 3.x mais recente do daemon. Para baixar uma versão específica, faça o seguinte:

- Para baixar uma versão anterior à versão 3.3.0, substitua 3.x pelo número da versão. Por exemplo, 2.1.0. Antes da versão 3.3.0, a única arquitetura disponível é arm64. Por exemplo, 2.1.0 e arm64.
- Para baixar uma versão posterior à versão 3.3.0, substitua 3.x pelo número da versão e arch pelo tipo de arquitetura.

Os ativos do X-Ray são replicados para buckets em todas as regiões compatíveis. Para usar o bucket mais próximo de você ou de seus AWS recursos, substitua a região nos links acima pela sua região.

```
https://s3.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-daemon/aws-xray-daemon-3.x.rpm
```

Amazon ECR

A partir da versão 3.2.0, o daemon pode ser encontrado no [Amazon ECR](#). Antes de extrair uma imagem, você deve [autenticar o cliente do Docker](#) no registro público do Amazon ECR.

Obtenha a tag da versão 3.x lançada mais recente executando o seguinte comando:

```
docker pull public.ecr.aws/xray/aws-xray-daemon:3.x
```

As versões anteriores ou alfa podem ser baixadas substituindo 3.x por alpha ou um número de versão específico. Não é recomendável usar uma imagem do daemon com uma tag alfa em um ambiente de produção.

Docker Hub

O daemon pode ser encontrado no [Docker Hub](#). Para baixar a versão 3.x lançada mais recente, execute o seguinte comando:

```
docker pull amazon/aws-xray-daemon:3.x
```

As versões anteriores do daemon podem ser lançadas substituindo 3.x pela versão desejada.

Verificar a assinatura de arquivamento do daemon

Os arquivos de assinatura GPG são incluídos para ativos do daemon compactados em arquivos ZIP. A chave pública está aqui: [aws-xray.gpg](#).

Você pode usar a chave pública para verificar se o arquivo morto ZIP do daemon é original e não modificado. Primeiro, importe a chave pública com [GnuPG](#).

Para importar a chave pública

1. Baixar a chave pública.

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
```

```
$ wget $BUCKETURL/xray-daemon/aws-xray.gpg
```

2. Importe a chave pública em seu token de autenticação.

```
$ 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
```

Use a chave importada para verificar a assinatura do arquivo morto ZIP do daemon.

Para verificar a assinatura de um arquivo morto

1. Baixe o arquivo morto e do arquivo de assinatura.

```
$ 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. Execute `gpg --verify` para verificar a assinatura.

```
$ 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
```

Observe o aviso sobre confiança. Uma chave só será confiável se você ou alguém em quem você confia a tiver assinado. Isso não significa que a assinatura é inválida, apenas que você não verificou a chave pública.

Execução do daemon

Execute o daemon localmente na linha de comando. Use a opção `-o` para executar em modo local e `-n` para definir a região.

```
~/Downloads$ ./xray -o -n us-east-2
```

Para obter instruções específicas da plataforma, consulte os tópicos a seguir:

- Linux (local): [Executar o daemon do X-Ray no Linux](#)
- Windows (local): [Executar o daemon do X-Ray-Ray no Windows](#)
- Elastic Beanstalk: [Executando o daemon X-Ray em AWS Elastic Beanstalk](#)
- Amazon EC2 — [Executando o daemon X-Ray na Amazon EC2](#)
- Amazon ECS: [Executar o daemon do X-Ray no Amazon ECS](#)

É possível personalizar ainda mais o comportamento do daemon usando as opções de linha de comando ou um arquivo de configuração. Para mais detalhes, consulte [Configurando o daemon AWS X-Ray](#).

Conceder permissão ao daemon para enviar dados ao X-Ray

O daemon X-Ray usa o AWS SDK para carregar dados de rastreamento no X-Ray e precisa de AWS credenciais com permissão para fazer isso.

Na Amazon EC2, o daemon usa automaticamente a função de perfil da instância. Para obter informações sobre as credenciais necessárias para executar o daemon localmente, consulte [Executar o aplicativo localmente](#).

Se você especificar as credenciais em mais de um local (arquivo de credenciais, perfil da instância ou variáveis de ambiente), a cadeia de fornecedores de SDK determinará quais credenciais serão usadas. Para obter mais informações sobre o fornecimento de credenciais para o SDK, consulte [Specifying Credentials](#) no Guia do desenvolvedor do AWS SDK para Go.

O usuário ou perfil do IAM a que as credenciais do daemon pertencem devem ter permissão para gravar dados no serviço em seu nome.

- Para usar o daemon na Amazon EC2, crie uma nova função de perfil de instância ou adicione a política gerenciada a uma existente.
- Para usar o daemon no Elastic Beanstalk, adicione a política gerenciada à função de perfil de instância padrão do Elastic Beanstalk.
- Para executar o daemon localmente, consulte [Executar o aplicativo localmente](#).

Para obter mais informações, consulte [Gerenciamento de identidade e acesso para AWS X-Ray](#).

Logs do daemon do X-Ray

O daemon gera informações sobre sua configuração atual e os segmentos para os quais ele envia. 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)
```

Por padrão, o daemon gera logs para STDOUT. Se você executar o daemon em segundo plano, use a opção de linha de comando `--log-file` ou um arquivo de configuração para definir o caminho do arquivo de log. Você também pode definir o nível de registro em log e desabilitar a rotação de logs. Para obter instruções, consulte [Configurando o daemon AWS X-Ray](#).

No Elastic Beanstalk, a plataforma define a localização dos logs do daemon. Para mais detalhes, consulte [Executando o daemon X-Ray em AWS Elastic Beanstalk](#).

Configurando o daemon AWS X-Ray

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

É possível usar opções de linha de comandos ou um arquivo de configuração para personalizar o comportamento do daemon do X-Ray. A maioria das opções estão disponíveis usando os dois métodos, mas alguns só estão disponíveis em arquivos de configuração e outros somente na linha de comando.

Para começar, a única opção necessária é `-n` ou `--region`, usada para definir a região que o daemon utilizará para enviar os dados de rastreamento ao X-Ray.


```
~/xray-daemon$ ./xray -n us-east-2
```

Se você estiver executando o daemon localmente, ou seja, não na Amazon EC2, poderá adicionar a `-o` opção de ignorar a verificação das credenciais do perfil da instância para que o daemon fique pronto mais rapidamente.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

As demais opções de linha de comandos permitem que você configure o registro em log, escute em uma porta diferente, limite a quantidade de memória que o daemon pode usar ou assumo um perfil para enviar dados de rastreamento a uma conta diferente.

Você pode passar um arquivo de configuração ao daemon para acessar as opções de configuração avançadas e executar tarefas, como limitar o número de chamadas simultâneas para o X-Ray, desabilitar a alternância de logs e enviar tráfego a um proxy.

Seções

- [Variáveis de ambiente compatíveis](#)
- [Usar as opções de linha de comando](#)
- [Usar um arquivo de configuração](#)

Variáveis de ambiente compatíveis

O daemon do X-Ray é compatível com as seguintes variáveis de ambiente:

- `AWS_REGION`: especifica a [Região da AWS](#) do endpoint de serviço do X-Ray.
- `HTTPS_PROXY`: especifica um endereço de proxy pelo qual o daemon carregará os segmentos. Isso pode ser os nomes de domínio DNS ou endereços IP e números de porta usados pelos servidores de proxy.

Usar as opções de linha de comando

Passe essas opções para o daemon quando você executá-lo localmente ou com um script de dados do usuário.

Opções de linha de comando

- `-b, --bind`: escutar os documentos de segmentos em uma porta UDP diferente.

```
--bind "127.0.0.1:3000"
```

Padrão: 2000.

- `-t, --bind-tcp`: escutar chamadas ao serviço X-Ray em uma porta TCP diferente.

```
-bind-tcp "127.0.0.1:3000"
```

Padrão: 2000.

- `-c, --config`: carregar um arquivo de configuração do caminho especificado.

```
--config "/home/ec2-user/xray-daemon.yaml"
```

- `-f, --log-file`: gerar logs de saída para o caminho do arquivo especificado.

```
--log-file "/var/log/xray-daemon.log"
```

- `-l, --log-level`: nível de log, do mais para o menos detalhado: dev, debug, info, warn, error, prod.

```
--log-level warn
```

Padrão: prod

- `-m, --buffer-memory`: alterar a quantidade de memória em MB que os buffers podem usar (3, no mínimo).

```
--buffer-memory 50
```

Padrão: 1% de memória disponível.

- `-o, --local-mode` — Não verifique, por EC2 exemplo, os metadados.
- `-r, --role-arn`: assumir o perfil do IAM especificado para carregar os segmentos para uma conta diferente.

```
--role-arn "arn:aws:iam::123456789012:role/xray-cross-account"
```

- `-a, --resource-arn` — Nome de recurso da Amazon (ARN) do AWS recurso que executa o daemon.
- `-p, --proxy-address` — Faça upload de segmentos AWS X-Ray por meio de um proxy. O protocolo do servidor proxy deve ser especificado.

```
--proxy-address "http://192.0.2.0:3000"
```

- `-n, --region`: enviar segmentos para o serviço X-Ray em uma região específica.
- `-v, --version` — Mostra a versão AWS X-Ray do daemon.
- `-h, --help`: mostrar a tela de ajuda.

Usar um arquivo de configuração

Também é possível usar um arquivo de formato YAML para configurar o daemon. Passe o arquivo de configuração para o daemon usando a opção `-c`.

```
~$ ./xray -c ~/xray-daemon.yaml
```

Opções do arquivo de configuração

- `TotalBufferSizeMB`: tamanho máximo do buffer em MB (3, no mínimo). Escolha 0 para usar 1% da memória de host.
- `Concurrency`— Número máximo de chamadas simultâneas AWS X-Ray para carregar documentos do segmento.
- `Region`— Envie segmentos para AWS X-Ray atendimento em uma região específica.
- `Socket`: configurar a vinculação do daemon.
 - `UDPEndress`: alterar a porta em que o daemon escuta.
 - `TCPAddress`: escutar [chamadas para o serviço do X-Ray](#) em uma porta TCP diferente.
- `Logging`: configurar o comportamento do registro em log.
 - `LogRotation`: definir como `false` para desabilitar a alternância de logs.
 - `LogLevel`: alterar o nível de log, do mais para o menos detalhado: `dev`, `debug`, `info` ou `prod`, `warn`, `error`, `prod`. O padrão é `prod`, que é equivalente a `info`.
 - `LogPath`: gerar logs para o caminho do arquivo especificado.
- `LocalMode`— Defina como `true` para ignorar a verificação dos metadados da EC2 instância.

- **ResourceARN**— Nome de recurso da Amazon (ARN) do AWS recurso que executa o daemon.
- **RoleARN**: assumir o perfil do IAM especificado faça carregar os segmentos para uma conta diferente.
- **ProxyAddress**— Faça upload de segmentos AWS X-Ray por meio de um proxy.
- **Endpoint**: alterar o endpoint de serviço do X-Ray para o qual o daemon envia documentos de segmentos.
- **NoVerifySSL**: desabilitar a verificação do certificado TLS.
- **Version**: versão do formato de arquivo de configuração do daemon. A versão do formato do arquivo é um campo obrigatório.

Example Xray-daemon.yaml

Esse arquivo de configuração altera a porta de escuta do daemon para 3000, desativa verificações de metadados da instância, define um perfil a ser usado para fazer upload de segmentos e altera opções de região e registro em log.

```
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
```

Executar o daemon do X-Ray localmente

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais

informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode executar o AWS X-Ray daemon localmente no Linux, macOS, Windows ou em um contêiner Docker. Execute o daemon para retransmitir os dados de rastreamento ao X-Ray quando você estiver desenvolvendo e testando seu aplicativo instrumentada. Baixe e extraia o daemon usando essas [instruções](#).

Ao ser executado localmente, o daemon pode ler as credenciais de um arquivo de credenciais do AWS SDK (.aws/credentials no seu diretório de usuário) ou de variáveis de ambiente. Para obter mais informações, consulte [Conceder permissão ao daemon para enviar dados ao X-Ray](#).

O daemon ouve dados UDP na porta 2000. Você pode alterar a porta e outras opções usando um arquivo de configuração e opções de linha de comando. Para obter mais informações, consulte [Configurando o daemon AWS X-Ray](#).

Executar o daemon do X-Ray no Linux

Você pode executar o daemon executável na linha de comando. Use a opção `-o` para executar em modo local e `-n` para definir a região.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Para executar o daemon em segundo plano, use `&`.

```
~/xray-daemon$ ./xray -o -n us-east-2 &
```

Encerre um processo de daemon em execução em segundo plano com `kill`.

```
~$ kill xray
```

Executar o daemon do X-Ray em um contêiner do Docker

Para executar o daemon localmente em um contêiner do Docker, salve o texto a seguir em um arquivo chamado `Dockerfile`. Baixe a [imagem de exemplo](#) completa no Amazon ECR. Para obter mais informações, consulte [Baixar o daemon](#).

Example Dockerfile: Amazon Linux

```
FROM amazonlinux
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

Crie a imagem do contêiner com o `docker build`.

```
~/xray-daemon$ docker build -t xray-daemon .
```

Execute a imagem em um contêiner com o `docker run`.

```
~/xray-daemon$ docker run \
  --attach STDOUT \
  -v ~/.aws/:/root/.aws/:ro \
  --net=host \
  -e AWS_REGION=us-east-2 \
  --name xray-daemon \
  -p 2000:2000/udp \
  xray-daemon -o
```

Este comando usa as seguintes opções:

- `--attach STDOUT`: visualizar a saída do daemon no terminal.
- `-v ~/.aws/:/root/.aws/:ro`— Dê ao contêiner acesso somente de leitura ao `.aws` diretório para permitir que ele leia suas credenciais do AWS SDK.
- `AWS_REGION=us-east-2`: definir a variável de ambiente da `AWS_REGION` para informar ao daemon a região a ser usada.
- `--net=host`: anexar o contêiner à rede do host. Os contêineres na rede host podem se comunicar entre si sem as portas de publicação.
- `-p 2000:2000/udp`: mapear a porta UDP 2000 no computador para a mesma porta no contêiner. Isso não é necessário para a comunicação entre contêineres na mesma rede, mas permite que você envie segmentos para o daemon [a partir da linha de comando](#) ou de um aplicativo que não está em execução no Docker.

- `--name xray-daemon`: nomear o contêiner como `xray-daemon` em vez de gerar um nome aleatório.
- `-o` (após o nome da imagem): anexar a opção `-o` ao ponto de entrada que executa o daemon dentro do contêiner. Essa opção faz com que o daemon seja executado no modo local para evitar que ele tente ler os metadados da EC2 instância Amazon.

Para interromper o daemon, use `docker stop`. Se você fizer alterações no `Dockerfile` e criar uma nova imagem, precisará excluir o contêiner existente antes de criar um outro com o mesmo nome. Use `docker rm` para excluir o contêiner.

```
$ docker stop xray-daemon
$ docker rm xray-daemon
```

Executar o daemon do X-Ray-Ray no Windows

Você pode executar o daemon executável na linha de comando. Use a opção `-o` para executar em modo local e `-n` para definir a região.

```
> .\xray_windows.exe -o -n us-east-2
```

Use um PowerShell script para criar e executar um serviço para o daemon.

Example PowerShell script - 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
```

Executar o daemon do X-Ray no OS X

Você pode executar o daemon executável na linha de comando. Use a opção `-o` para executar em modo local e `-n` para definir a região.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2
```

Para executar o daemon em segundo plano, use `&`.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2 &
```

Use `nohup` para evitar que o daemon seja encerrado quando o terminal for fechado.

```
~/xray-daemon$ nohup ./xray_mac &
```

Executando o daemon X-Ray em AWS Elastic Beanstalk

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Para retransmitir dados de rastreamento do seu aplicativo para AWS X-Ray, você pode executar o daemon X-Ray nas instâncias da Amazon do seu ambiente Elastic Beanstalk. EC2 Para obter

uma lista de plataformas compatíveis, consulte [Configurar depuração do AWS X-Ray](#) no Guia do desenvolvedor do AWS Elastic Beanstalk .

 Note

O daemon usa seu perfil de instância do ambiente para permissões. Para obter instruções sobre como adicionar permissões ao perfil de instância do Elastic Beanstalk, consulte [Conceder permissão ao daemon para enviar dados ao X-Ray](#).

As plataformas do Elastic Beanstalk fornecem uma opção de configuração que você pode definir para executar o daemon automaticamente. Você pode habilitar o daemon em um arquivo de configuração em seu código-fonte ou selecionar uma opção no console do Elastic Beanstalk. Quando você habilita a opção de configuração, o daemon é instalado na instância e é executado como um serviço.

A versão incluída nas plataformas do Elastic Beanstalk pode não ser a versão mais recente. Consulte o [tópico Plataformas suportadas](#) para descobrir a versão do daemon que está disponível para a configuração da sua plataforma.

O Elastic Beanstalk não fornece o daemon do X-Ray na plataforma Docker de vários contêineres (Amazon ECS).

Usar a integração entre do X-Ray com o Elastic Beanstalk para executar o daemon do X-Ray

Use o console para ativar a integração do X-Ray ou configure-o no código-fonte do aplicativo com um arquivo de configuração.

Como habilitar o daemon do X-Ray no console do Elastic Beanstalk

1. Abra o [console do Elastic Beanstalk](#).
2. Navegue até o [console de gerenciamento](#) do seu ambiente.
3. Escolher configuração.
4. Escolha Software Settings.
5. Para o X-Ray daemon, escolha Enabled.
6. Escolha Aplicar.

Você pode incluir um arquivo de configuração em seu código-fonte para tornar sua configuração portátil entre ambientes.

Example `.ebextensions/xray-daemon.config`

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

O Elastic Beanstalk transmite um arquivo de configuração ao daemon e emite logs para um local padrão.

Nas Plataformas do Windows Server

- Arquivo de configuração: `C:\Program Files\Amazon\XRay\cfg.yaml`
- Logs: `c:\Program Files\Amazon\XRay\logs\xray-service.log`

Nas Plataformas Linux

- Arquivo de configuração: `/etc/amazon/xray/cfg.yaml`
- Logs: `/var/log/xray/xray.log`

O Elastic Beanstalk fornece ferramentas para extrair registros de instâncias da linha de comando ou. Console de gerenciamento da AWS Você pode instruir o Elastic Beanstalk a incluir os logs do daemon do X-Ray adicionando uma tarefa com um arquivo de configuração.

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:\Program Files\Amazon\XRay\logs\xray-service.log
```

Consulte [Visualização de registros das AWS Elastic Beanstalk instâncias da Amazon do seu ambiente Elastic Beanstalk no EC2](#) Guia do desenvolvedor para obter mais informações.

Baixar e executar o daemon do X-Ray manualmente (avançado)

Se o daemon do X-Ray não estiver disponível para a configuração de sua plataforma, você poderá baixá-lo pelo Amazon S3 e executá-lo com um arquivo de configuração.

Use um arquivo de configuração do Elastic Beanstalk para baixar e executar o daemon.

Example .ebextensions/xray.config: Linux

```
commands:
  01-stop-tracing:
    command: yum remove -y xray
    ignoreErrors: true
  02-copy-tracing:
    command: curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
  03-start-tracing:
    command: yum install -y /home/ec2-user/xray.rpm

files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      /var/log/xray/xray.log
  "/etc/amazon/xray/cfg.yaml" :
    mode: "000644"
    owner: root
    group: root
    content: |
      Logging:
        LogLevel: "debug"
      Version: 2
```

Example .ebextensions/xray.config – Windows Server

```

container_commands:
  01-execute-config-script:
    command: Powershell.exe -ExecutionPolicy Bypass -File c:\\temp\\installDaemon.ps1
    waitAfterCompletion: 0

files:
  "c:/temp/installDaemon.ps1":
    content: |
      if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ) {
        sc.exe stop AWSXRayDaemon
        sc.exe delete AWSXRayDaemon
      }

      $targetLocation = "C:\Program Files\Amazon\XRay"
      if ((Test-Path $targetLocation) -eq 0) {
        mkdir $targetLocation
      }

      $zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
      $zipPath = "$targetLocation\$zipFileName"
      $destPath = "$targetLocation\aws-xray-daemon"
      if ((Test-Path $destPath) -eq 1) {
        Remove-Item -Recurse -Force $destPath
      }

      $daemonPath = "$destPath\xray.exe"
      $daemonLogPath = "$targetLocation\xray-daemon.log"
      $url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/
xray-daemon/aws-xray-daemon-windows-service-3.x.zip"

      Invoke-WebRequest -Uri $url -OutFile $zipPath
      Add-Type -Assembly "System.IO.Compression.FileSystem"
      [io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

      New-Service -Name "AWSXRayDaemon" -StartupType Automatic -BinaryPathName
      "`"$daemonPath`" -f "`"$daemonLogPath`""
      sc.exe start AWSXRayDaemon
      encoding: plain
  "c:/Program Files/Amazon/ElasticBeanstalk/config/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root

```

```
content: |  
C:\Program Files\Amazon\XRay\xray-daemon.log
```

Esses exemplos também adicionam o arquivo de log do daemon à tarefa de logs finais do Elastic Beanstalk, para que ele esteja incluído quando você solicitar logs com o console ou com a interface de linha de comandos (CLI do EB) do Elastic Beanstalk.

Executando o daemon X-Ray na Amazon EC2

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode executar o daemon X-Ray nos seguintes sistemas operacionais na Amazon: EC2

- Amazon Linux
- Ubuntu
- Windows Server (2012 R2 e mais recente)

Use um perfil de instância para conceder ao daemon permissão para fazer upload de dados de rastreamento no X-Ray. Para obter mais informações, consulte [Conceder permissão ao daemon para enviar dados ao X-Ray](#).

Use um script de dados de usuário para executar o daemon automaticamente quando você iniciar a instância.

Example Script de dados do usuário – Linux

```
#!/bin/bash  
curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
```

```
yum install -y /home/ec2-user/xray.rpm
```

Example Script de dados do usuário – 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>
```

Executar o daemon do X-Ray no Amazon ECS

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-

Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

No Amazon ECS, crie uma imagem do Docker que executa o daemon do X-Ray, carregue-a em um repositório de imagens do Docker e, em seguida, implante-a no cluster do Amazon ECS. Você pode usar os mapeamentos de porta e as configurações de modo de rede em seu arquivo de definição de tarefa para permitir que o aplicativo se comunique com o contêiner do daemon.

Usar a imagem oficial do Docker da

O X-Ray fornece uma [imagem de contêiner](#) do Docker no Amazon ECR que você pode implantar com o aplicativo. Para obter mais informações, consulte [Baixar o daemon](#).

Exemplo definição da tarefa

```
{
  "name": "xray-daemon",
  "image": "amazon/aws-xray-daemon",
  "cpu": 32,
  "memoryReservation": 256,
  "portMappings" : [
    {
      "hostPort": 0,
      "containerPort": 2000,
      "protocol": "udp"
    }
  ]
}
```

Criar e compilar uma imagem do Docker

Para a configuração personalizada, pode ser necessário definir sua própria imagem do Docker.

Adicione políticas gerenciadas no perfil da tarefa para conceder ao daemon permissão para carregar dados de rastreamento no X-Ray. Para obter mais informações, consulte [Conceder permissão ao daemon para enviar dados ao X-Ray](#).

Use um dos seguintes Dockerfiles para criar uma imagem que executa o daemon.

Example Dockerfile: Amazon Linux

```
FROM amazonlinux
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

Note

Os sinalizadores `-t` e `-b` são necessários para especificar um endereço de vinculação para ouvir o loopback de um ambiente com vários contêineres.

Example Dockerfile: Ubuntu

Para derivados do Debian, você também precisa instalar certificados da autoridade de certificação (CA) para evitar problemas ao baixar o instalador.

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y --force-yes --no-install-recommends apt-
transport-https curl ca-certificates wget && apt-get clean && apt-get autoremove && rm
-rf /var/lib/apt/lists/*
RUN wget https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-
xray-daemon-3.x.deb
RUN dpkg -i aws-xray-daemon-3.x.deb
ENTRYPOINT ["/usr/bin/xray", "--bind=0.0.0.0:2000", "--bind-tcp=0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

Na definição da tarefa, a configuração vai depender do modo de rede que você usa. A rede bridge é o padrão e pode ser usado na sua VPC padrão. Em uma rede de ponte, defina a variável de ambiente `AWS_XRAY_DAEMON_ADDRESS` para informar ao X-Ray SDK a qual porta de contêiner deve ser feita referência e defina a porta do host. Por exemplo, você poderia publicar a porta UDP 2000 e criar um link do contêiner do aplicativo para o contêiner do daemon.

Exemplo definição da tarefa

```
{
  "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"
  ]
}
```

Se você executar o seu cluster na sub-rede privada de uma VPC, poderá usar o [awsvpc modo de rede](#) para anexar uma interface de rede elástica (ENI) aos seus contêineres. Isso permite que você evite o uso de links. Omita a porta host nos mapeamentos de porta, o link e a variável de ambiente `AWS_XRAY_DAEMON_ADDRESS`.

Exemplo Definição de tarefa da VPC

```
{
  "family": "scorekeep",
  "networkMode": "awsvpc",
  "containerDefinitions": [
    {
      "name": "xray-daemon",
      "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
      "cpu": 32,
      "memoryReservation": 256,
      "portMappings": [
        {
          "containerPort": 2000,
          "protocol": "udp"
        }
      ]
    },
    {
      "name": "scorekeep-api",
      "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
      "cpu": 192,
      "memoryReservation": 512,
      "environment": [
        { "name": "AWS_REGION", "value": "us-east-2" },
        { "name": "NOTIFICATION_TOPIC", "value": "arn:aws:sns:us-east-2:123456789012:scorekeep-notifications" }
      ],
      "portMappings": [
        {
          "containerPort": 5000
        }
      ]
    }
  ]
}
```

Configurar opções da linha de comandos no console do Amazon ECS

As opções da linha de comando substituem qualquer valor conflitante no arquivo de configuração da imagem. As opções da linha de comando normalmente são usadas para testes locais, mas também

podem ser usadas por conveniência ao definir variáveis de ambiente ou para controlar o processo de inicialização.

Ao adicionar opções da linha de comando, você está atualizando o Docker CMD que é transmitido para o contêiner. Para obter mais informações, consulte a [Referência de execução do Docker](#).

Como definir uma opção da linha de comando

1. Abra o console clássico do Amazon ECS em <https://console.aws.amazon.com/ecs/>.
2. Na barra de navegação, selecione a região que contém a definição de tarefa.
3. No painel de navegação, selecione Definições de tarefas.
4. Na página Task Definitions, selecione a caixa à esquerda da definição de tarefa a ser revisada e escolha Create new revision.
5. Na página Create new revision of Task Definition (Criar revisão da definição de tarefa), selecione o contêiner.
6. Na seção ENVIRONMENT (AMBIENTE), adicione a lista separada por vírgulas das opções da linha de comando ao campo Command (Comando).
7. Selecione Atualizar.
8. Verifique as informações e escolha Create.

O exemplo a seguir mostra como escrever uma lista separada por vírgulas da opção da linha de comando para a opção RoleARN. A opção RoleARN assume o perfil do IAM especificado para carregar segmentos em uma conta diferente.

Example

```
--role-arn, arn:aws:iam::123456789012:role/xray-cross-account
```

Para saber mais sobre as opções de linha de comando disponíveis no X-Ray, consulte [Configurando o AWS X-Ray Daemon](#).

Integrando AWS X-Ray com outros Serviços da AWS

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Muitos Serviços da AWS oferecem níveis variados de integração com o X-Ray, incluindo amostragem e adição de cabeçalhos às solicitações recebidas, execução do daemon X-Ray e envio automático de dados de rastreamento para o X-Ray. A integração com o X-Ray pode incluir o seguinte:

- Instrumentação ativa: realiza amostragens e instrumenta as solicitações de entrada.
- Instrumentação passiva: instrumenta as solicitações que foram amostradas por outro serviço.
- Rastreamento de solicitação: adiciona um cabeçalho de rastreamento a todas as solicitações de entrada e o propaga subsequentemente.
- Ferramentas: executa o daemon do X-Ray para receber segmentos do X-Ray SDK;

Note

O X-Ray SDKs inclui plug-ins para integração adicional com Serviços da AWS. Por exemplo, você pode usar o plug-in do Elastic Beanstalk do X-Ray SDK para Java para adicionar informações sobre o ambiente do Elastic Beanstalk no qual o aplicativo está sendo executada, incluindo o nome e o ID do ambiente.

Aqui estão alguns exemplos Serviços da AWS que estão integrados ao X-Ray:

- [AWS Distro for OpenTelemetry \(ADOT\)](#) — Com o ADOT, os engenheiros podem instrumentar seus aplicativos uma vez e enviar métricas e rastreamentos correlacionados para várias AWS

soluções de monitoramento, incluindo Amazon CloudWatch, Amazon Service e AWS X-Ray Amazon OpenSearch Managed Service for Prometheus.

- [AWS Lambda](#)— Instrumentação ativa e passiva das solicitações recebidas em todos os tempos de execução. AWS Lambda adiciona dois nós ao seu mapa de rastreamento, um para o AWS Lambda serviço e outro para a função. Quando você ativa a instrumentação, AWS Lambda também executa o daemon X-Ray em tempos de execução Java e Node.js para uso com o X-Ray SDK.
- [Amazon API Gateway](#): instrumentação ativa e passiva. O API Gateway usa regras de amostragem para determinar quais solicitações serão registradas e adiciona um nó para o estágio do gateway ao seu mapa de serviço.
- [AWS Elastic Beanstalk](#): ferramentas. O Elastic Beanstalk inclui o daemon do X-Ray nas seguintes plataformas:
 - Java SE: 2.3.0 e configurações posteriores
 - Tomcat: 2.4.0 e configurações posteriores
 - Node.js: 3.2.0 e configurações posteriores
 - Windows Server: todas as configurações foram lançadas após 9 de dezembro de 2016, com a exceção do Windows Server Core.

É possível usar o console do Elastic Beanstalk para instruir o Elastic Beanstalk a executar o daemon nessas plataformas ou usar a opção `XRayEnabled` no namespace `aws:elasticbeanstalk:xray`.

- [Elastic Load Balancing](#): rastreamento de solicitações em Application Load Balancers. O Application Load Balancer adiciona o ID de rastreamento ao cabeçalho de solicitação antes de enviá-lo para um grupo de destino.
- [Amazon EventBridge](#) — Instrumentação passiva. Se um serviço que publica eventos EventBridge for instrumentado com o X-Ray SDK, os destinos do evento receberão o cabeçalho de rastreamento e poderão continuar a propagar o ID de rastreamento original.
- [Amazon Simple Notification Service](#): instrumentação passiva. Caso um publicador do Amazon SNS rastreie seus clientes com o X-Ray SDK, os assinantes podem recuperar o cabeçalho de rastreamento e continuar a propagar o rastreamento original do publicador com o mesmo ID de rastreamento.
- [Amazon Simple Queue Service](#): instrumentação passiva. Caso um serviço rastreie solicitações usando o X-Ray SDK, o Amazon SQS pode enviar o cabeçalho de rastreamento e continuar a propagar o rastreamento original do remetente para o consumidor com um ID de rastreamento consistente.

- [Amazon Bedrock AgentCore](#) — AgentCore oferece suporte ao rastreamento distribuído por meio da integração com o X-Ray, permitindo que você acompanhe as solicitações à medida que elas fluem pelos aplicativos do seu agente. Ao habilitar a observabilidade de seus AgentCore recursos, você pode propagar o contexto de rastreamento entre os limites do serviço e obter visibilidade do desempenho de seus agentes e ferramentas de IA.

Escolha um dos tópicos a seguir para explorar o conjunto completo de opções integradas Serviços da AWS.

Tópicos

- [Amazon Bedrock AgentCore e AWS X-Ray](#)
- [Amazon Elastic Compute Cloud e AWS X-Ray](#)
- [Amazon SNS e AWS X-Ray](#)
- [Amazon SQS e AWS X-Ray](#)
- [Amazon S3 e AWS X-Ray](#)
- [AWS Distro para OpenTelemetry e AWS X-Ray](#)
- [Rastrear as alterações da configuração de criptografia do X-Ray com o AWS Config](#)
- [AWS AppSync e AWS X-Ray](#)
- [Compatibilidade com o rastreamento ativo do Amazon API Gateway para o AWS X-Ray](#)
- [Amazon EC2 e AWS App Mesh](#)
- [AWS App Runner e X-Ray](#)
- [Registro em log de chamadas de API do X-Ray com o AWS CloudTrail](#)
- [Integração ao CloudWatch para o X-Ray](#)
- [AWS Elastic Beanstalk and AWS X-Ray](#)
- [Elastic Load Balancing e AWS X-Ray](#)
- [Amazon EventBridge e AWS X-Ray](#)
- [AWS Lambda and AWS X-Ray](#)
- [AWS Step Functions e AWS X-Ray](#)

Amazon Bedrock AgentCore e AWS X-Ray

O Amazon Bedrock AgentCore se integra AWS X-Ray para fornecer recursos de rastreamento distribuído para seus agentes e ferramentas de IA. Essa integração permite rastrear as solicitações

à medida que elas fluem pelas aplicações dos seus agentes, ajudando você a identificar gargalos de desempenho e solucionar problemas.

AgentCore oferece suporte ao rastreamento distribuído por meio da integração com o X-Ray, permitindo monitorar o desempenho de seus agentes e ferramentas de IA. Ao habilitar a observabilidade para seus AgentCore recursos, você pode propagar o contexto de rastreamento entre os limites do serviço e obter visibilidade de como seus agentes interagem com outros AWS serviços. Para obter mais informações, consulte [Amazon Bedrock AgentCore](#).

AgentCore suporta os seguintes recursos do X-Ray:

- Propagação do contexto de rastreamento para serviços downstream
- Instrumentação personalizada usando o AWS Distro for OpenTelemetry (ADOT) SDK

Configurando o X-Ray com AgentCore

Para usar o X-Ray com AgentCore, você precisa ativar a Pesquisa de CloudWatch Transações em sua AWS conta. Essa é uma configuração única que permite enviar dados de rastreamento AgentCore para o X-Ray. Para obter mais informações, consulte [Habilitar o Transaction Search](#).

Para obter mais informações sobre como configurar a observabilidade para AgentCore, consulte [Adicionar observabilidade ao seu AgentCore agente ou ferramenta Amazon Bedrock](#).

Usando cabeçalhos de rastreamento com AgentCore

AgentCore suporta o formato de cabeçalho de rastreamento X-Ray para rastreamento distribuído. Você pode incluir o `X-Amzn-Trace-Id` cabeçalho em suas solicitações AgentCore para manter o contexto de rastreamento entre os limites do serviço.

Amazon Elastic Compute Cloud e AWS X-Ray

Você pode instalar e executar o daemon do X-Ray em uma instância do Amazon EC2 com um script de dados do usuário. Para obter instruções, consulte [Executando o daemon X-Ray na Amazon EC2](#).

Use um perfil de instância para conceder ao daemon permissão para fazer upload de dados de rastreamento no X-Ray. Para obter mais informações, consulte [Conceder permissão ao daemon para enviar dados ao X-Ray](#).

Amazon SNS e AWS X-Ray

Você pode usar o AWS X-Ray com o Amazon Simple Notification Service (Amazon SNS) para rastrear e analisar solicitações à medida que elas passam pelos tópicos do SNS até os [serviços de assinatura compatíveis com o SNS](#). Use o rastreamento do X-Ray com o Amazon SNS para analisar latências em mensagens e serviços de back-end, como quanto tempo uma solicitação permanece em um tópico e quanto tempo leva para entregar a mensagem a cada uma das assinaturas do tópico. O Amazon SNS permite o rastreamento do X-Ray para tópicos comuns e FIFO.

Se você publicar em um tópico do Amazon SNS usando um serviço que já está instrumentado com o X-Ray, o Amazon SNS passará o contexto de rastreamento do publicador para os assinantes. Além disso, você pode ativar o rastreamento ativo para enviar dados de segmentos sobre suas assinaturas do Amazon SNS ao X-Ray para mensagens publicadas de um cliente do SNS instrumentado. [Ative o rastreamento ativo](#) para um tópico do Amazon SNS usando o console do Amazon SNS ou usando a CLI ou a API do Amazon SNS. Consulte [Instrumentar sua aplicação](#) para obter mais informações sobre como instrumentar clientes do SNS.

Configurar o rastreamento ativo do Amazon SNS

Você pode usar o console do Amazon SNS ou a AWS CLI ou o SDK para configurar o rastreamento ativo do Amazon SNS.

Ao usar o console do Amazon SNS, o Amazon SNS tenta criar as permissões necessárias para que o SNS chame o X-Ray. A tentativa poderá ser rejeitada se você não tiver permissões suficientes para modificar as políticas de recursos do X-Ray. Para obter mais informações sobre o controle de acesso do Amazon SNS, consulte [Identity and access management in Amazon SNS](#) e [Casos de exemplo para o controle de acesso do Amazon SNS](#) no Guia do desenvolvedor do Amazon Simple Notification Service. Para obter mais informações sobre como ativar o rastreamento ativo usando o console do Amazon SNS, consulte [Habilitar o rastreamento ativo em um tópico do Amazon SNS](#) no Guia do desenvolvedor do Amazon Simple Notification Service.

Ao utilizar a AWS CLI ou o SDK para ativar o rastreamento ativo, você deve configurar manualmente as permissões usando políticas baseadas em recursos. Use [PutResourcePolicy](#) para configurar o X-Ray com a política baseada em recursos necessária para permitir que o Amazon SNS envie rastreamentos ao X-Ray.

Exemplo Exemplo de política baseada em recursos do X-Ray para rastreamento ativo do Amazon SNS

Este exemplo de documento de política especifica as permissões que o Amazon SNS precisa para enviar dados de rastreamento ao X-Ray:

```
{
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "SNSAccess",
      Effect: Allow,
      Principal: {
        Service: "sns.amazonaws.com",
      },
      Action: [
        "xray:PutTraceSegments",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets"
      ],
      Resource: "*",
      Condition: {
        StringEquals: {
          "aws:SourceAccount": "account-id"
        },
        StringLike: {
          "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
        }
      }
    }
  ]
}
```

Use a CLI para criar uma política baseada em recursos que conceda permissões ao Amazon SNS para enviar dados de rastreamento ao X-Ray:

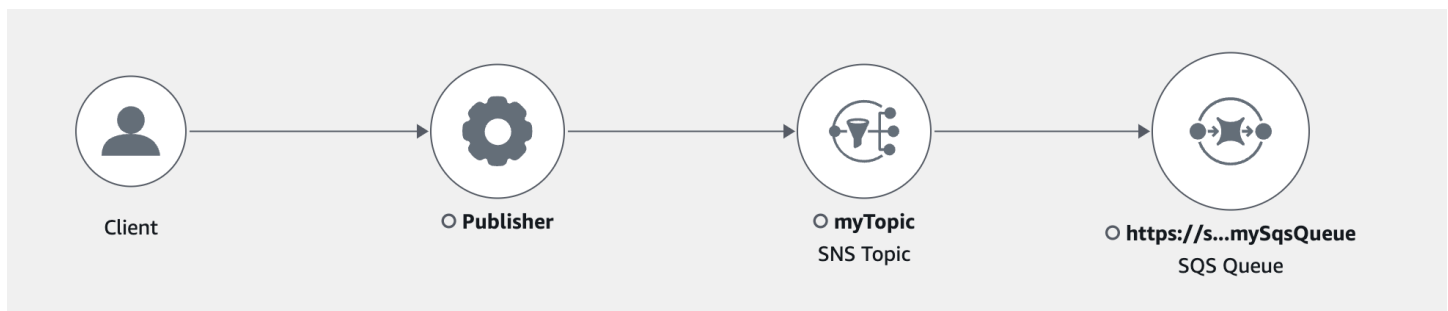
```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
'{ "Version": "2012-10-17",      "Statement": [ { "Sid": "SNSAccess",
"Effect": "Allow", "Principal": { "Service": "sns.amazonaws.com" }, "Action":
[ "xray:PutTraceSegments", "xray:GetSamplingRules", "xray:GetSamplingTargets" ],
"Resource": "*", "Condition": { "StringEquals": { "aws:SourceAccount": "account-
```

```
id" }, "StringLike": { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name" } } ] }'
```

Para usar esses exemplos, substitua *partition*, *region*, *account-id* e *topic-name* por sua partição, região, ID da conta e nome de tópico do Amazon SNS específicos da AWS. Para conceder permissão a todos os tópicos do Amazon SNS para enviar dados de rastreamento ao X-Ray, substitua o nome do tópico por `*`.

Visualize rastreamentos de publicadores e assinantes do Amazon SNS no console do X-Ray

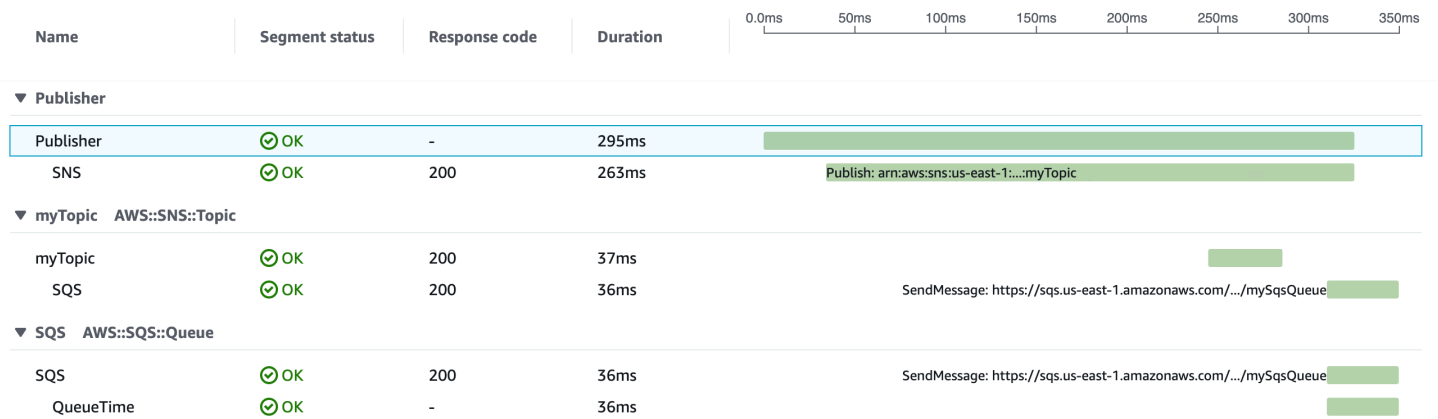
Use o console do X-Ray para visualizar um mapa de serviço, um mapa de rastreamento e detalhes de rastreamento que exibem uma visão conectada dos publicadores e assinantes do Amazon SNS. Quando o rastreamento ativo do Amazon SNS é ativado para um tópico, o mapa do serviço do X-Ray e o mapa de rastreamento exibem nós conectados para os publicadores do Amazon SNS, o tópico do Amazon SNS e os assinantes subsequentes:



Depois que você escolhe um rastreamento que abrange um publicador e assinante do Amazon SNS, a página de detalhes do rastreamento X-Ray exibe um mapa de rastreamento e uma linha do tempo de segmento.

Exemplo Exemplo de linha do tempo com publicador e assinante do Amazon SNS

Este exemplo mostra uma linha do tempo que inclui um publicador do Amazon SNS que envia uma mensagem a um tópico do Amazon SNS, o qual é processado por um assinante do Amazon SQS.

Segments Timeline [Info](#)

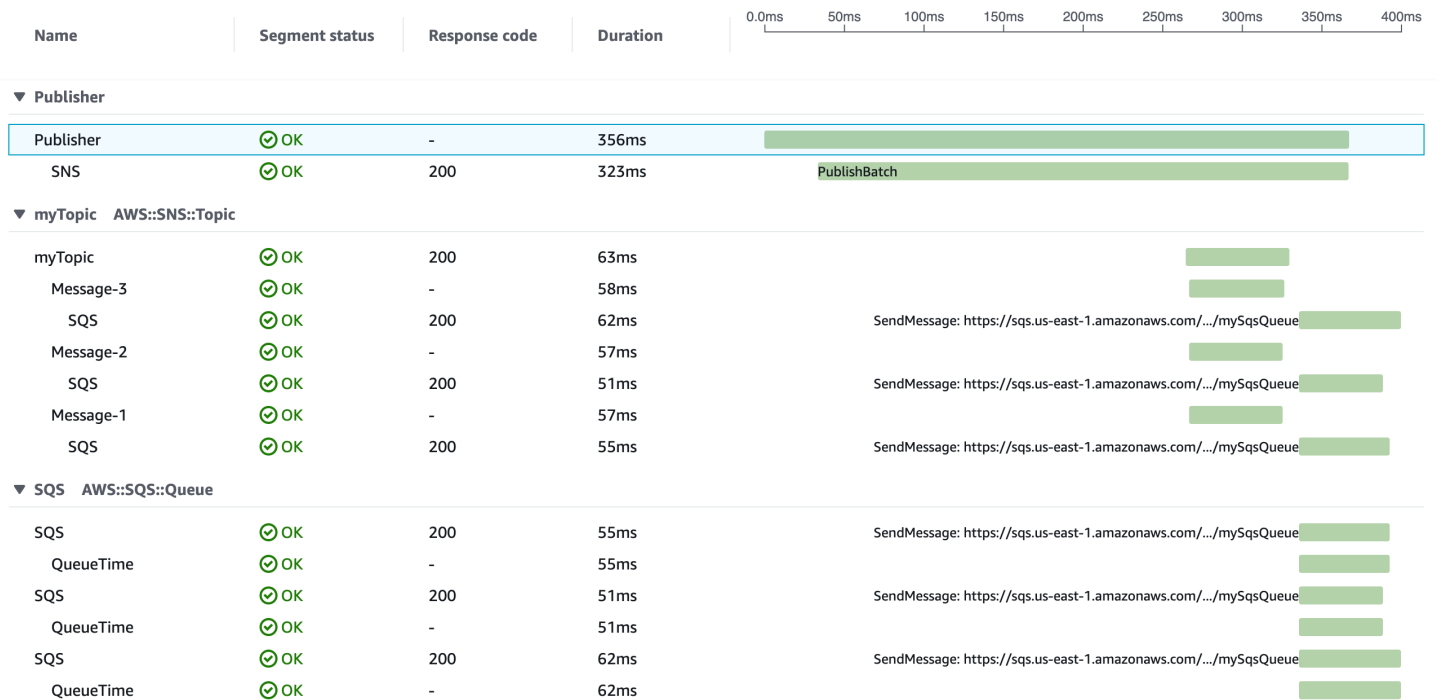
O exemplo de linha do tempo acima fornece detalhes sobre o fluxo de mensagens do Amazon SNS:

- O segmento SNS representa a duração do percurso de ida e volta da chamada de API Publish do cliente.
- O segmento myTopic representa a latência da resposta do Amazon SNS à solicitação de publicação.
- O subsegmento SQS representa o tempo de ida e volta que o Amazon SNS leva para publicar a mensagem em uma fila do Amazon SQS.
- O tempo entre o segmento myTopic e o subsegmento SQS representa o tempo que a mensagem passa no sistema do Amazon SNS.

Example Exemplo de linha do tempo com mensagens em lote do Amazon SNS

Se várias mensagens do Amazon SNS forem agrupadas em um único rastreamento, a linha do tempo do segmento exibirá segmentos que representam cada mensagem processada.

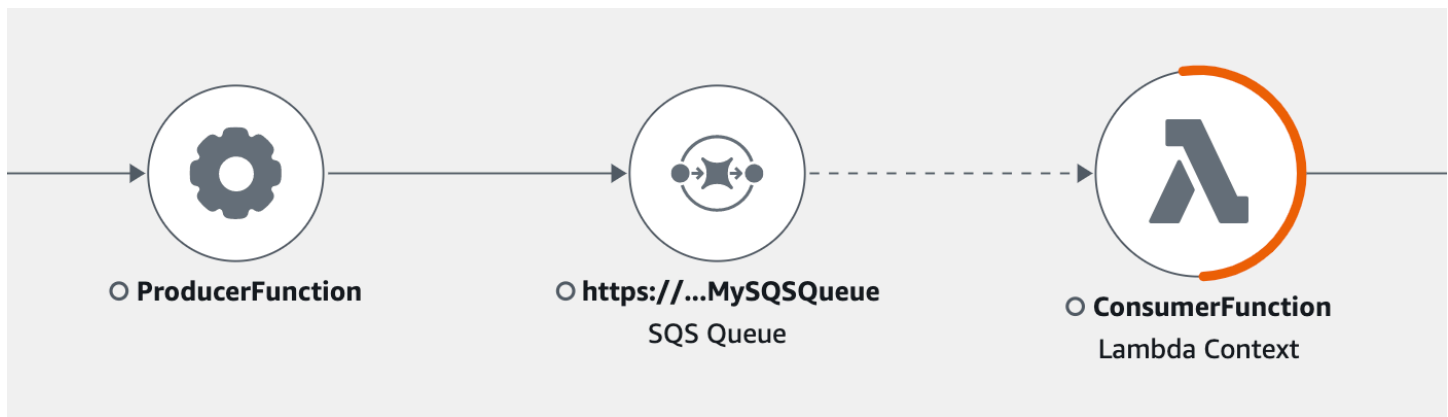
Segments Timeline [Info](#)



Amazon SQS e AWS X-Ray

O AWS X-Ray integra-se ao Amazon Simple Queue Service (Amazon SQS) para rastrear mensagens que são passadas por uma fila do Amazon SQS. Caso um serviço rastreie solicitações usando o X-Ray SDK, o Amazon SQS pode enviar o cabeçalho de rastreamento e continuar a propagar o rastreamento original do remetente para o consumidor com um ID de rastreamento consistente. Essa continuidade permite que os usuários rastreiem, analisem, e depurem em serviços de downstream.

O AWS X-Ray é compatível com o rastreamento de aplicações orientados a eventos usando o Amazon SQS e o AWS Lambda. Use o console do CloudWatch para ver uma visualização conectada de cada solicitação à medida que ela é enfileirada com o Amazon SQS e processada por uma função do Lambda subsequente. Os rastreamentos dos produtores de mensagens precedentes são automaticamente vinculados aos rastreamentos dos nós consumidores subsequentes do Lambda, criando uma visão completa da aplicação. Para obter mais informações, consulte [Rastrear aplicações orientadas a eventos](#).



O Amazon SQS permite a seguinte instrumentação de cabeçalho de rastreamento:

- **Cabeçalho HTTP padrão:** o X-Ray SDK preenche automaticamente o cabeçalho de rastreamento como um cabeçalho HTTP quando você chama o Amazon SQS por meio do SDK da AWS. O cabeçalho de rastreamento padrão é carregado por `X-Amzn-Trace-Id` e corresponde a todas as mensagens incluídas em uma solicitação [SendMessage](#) ou [SendMessageBatch](#). Para saber mais sobre o cabeçalho HTTP padrão, consulte [Cabeçalho de rastreamento](#).
- **Atributo de sistema `AWSTraceHeader`:** o `AWSTraceHeader` é um [atributo de sistema de mensagens](#) reservado pelo Amazon SQS para transportar o cabeçalho de rastreamento do X-Ray com mensagens na fila. O `AWSTraceHeader` está disponível para uso mesmo quando a instrumentação automática por meio do X-Ray SDK não está; por exemplo, ao criar um SDK de rastreamento para uma nova linguagem. Quando ambas as instrumentações de cabeçalho são definidas, o atributo do sistema de mensagens substitui o cabeçalho de rastreamento HTTP.

Quando executado no Amazon EC2, o Amazon SQS permite o processamento de uma mensagem por vez. Isso se aplica quando em execução em um host on-premises e ao usar serviços de contêiner, como o AWS Fargate, o Amazon ECS ou o AWS App Mesh.

O cabeçalho de rastreamento é excluído das cotas de atributos de mensagem e tamanho de mensagem do Amazon SQS. A habilitação do rastreamento do X-Ray não excederá suas cotas do Amazon SQS. Para saber mais sobre cotas da AWS, consulte [Cotas do Amazon SQS](#).

Enviar o cabeçalho de rastreamento HTTP

Os componentes do remetente no Amazon SQS podem enviar o cabeçalho de rastreamento automaticamente por meio da chamada [SendMessageBatch](#) ou [SendMessage](#). Quando os clientes de SDK da AWS são instrumentados, eles podem ser rastreados automaticamente por todas as linguagens compatíveis por meio do X-Ray SDK. Os Serviços da AWS rastreados e os recursos que

you access these services (for example, an Amazon S3 bucket or an Amazon SQS queue) are displayed as downstream nodes in the service map in the X-Ray console.

To learn how to trace calls to the AWS SDK with the language of your preference, consult the following topics in the compatible SDKs:

- Go – [Rastreando chamadas AWS do SDK com o X-Ray SDK for Go](#)
- Java: [Rastreando chamadas AWS do SDK com o X-Ray SDK for Java](#)
- Node.js – [Rastreando chamadas AWS do SDK com o X-Ray SDK para Node.js](#)
- Python – [Rastreando chamadas AWS do SDK com o X-Ray SDK para Python](#)
- Ruby – [Rastreando chamadas AWS do SDK com o X-Ray SDK for Ruby](#)
- .NET – [Rastreando chamadas AWS do SDK com o X-Ray SDK for .NET](#)

Recuperar o cabeçalho de rastreamento e recuperar o contexto de rastreamento

If you are using a downstream consumer of Lambda, the propagation of the context of the trace will be automatic. To give continuity to the propagation of context with the clients of Amazon SQS, you will have to instrument manually the transfer for the receptor component.

Existem três etapas principais para recuperar o contexto de rastreamento:

- Receive the message from the queue for the attribute `AWSTraceHeader` calling the API [ReceiveMessage](#).
- Recover the trace header of the attribute.
- Recover the ID of the trace of the header. Optionally, add more metrics to the segment.

See the following example of implementation written with the X-Ray SDK for Java.

Example : Recover the trace header and recover the context of the trace

```
// 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 e AWS X-Ray

O AWS X-Ray se integra ao Amazon S3 para rastrear solicitações precedentes para atualizar os buckets do S3 da aplicação. Se um serviço rastreia solicitações usando o X-Ray SDK, o Amazon S3 pode enviar os cabeçalhos de rastreamento para assinantes de eventos subsequentes, como o AWS Lambda, o Amazon SQS e o Amazon SNS. O X-Ray permite mensagens de rastreamento para notificações de eventos do Amazon S3.

É possível usar o X-Ray para visualizar o mapa de conexões entre o Amazon S3 e outros serviços que a aplicação usa. Também é possível usar o console para visualizar métricas como a latência média e as taxas de falha. Para obter mais informações sobre como usar o console do X-Ray, consulte [Usar o console do X-Ray](#).

O Amazon S3 permite a instrumentação de cabeçalho http padrão. O X-Ray SDK preenche automaticamente o cabeçalho de rastreamento como um cabeçalho HTTP quando você chama o Amazon S3 por meio do SDK da AWS. O cabeçalho de rastreamento padrão é transportado por X-Amzn-Trace-Id. Para saber mais sobre o rastreamento de cabeçalhos, consulte [Cabeçalho de rastreamento](#) na página de conceitos. A propagação do contexto de rastreamento do Amazon S3 pode ser utilizada pelos seguintes assinantes: Lambda, SQS e SNS. Como o SQS e o SNS não emitem dados de segmento por conta própria, eles não aparecerão em seu rastreamento ou mapa de serviço quando acionados pelo S3, mesmo que propaguem o cabeçalho de rastreamento para serviços subsequentes.

Configurar notificações de eventos do Amazon S3

Com o recurso de notificação do Amazon S3, você recebe notificações quando certos eventos ocorrem no bucket. Essas notificações podem então ser propagadas para os seguintes destinos na aplicação:

- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- AWS Lambda

Para obter uma lista de eventos aceitos, consulte os [tipos de eventos compatíveis no Guia do usuário do Amazon S3](#).

Amazon SNS e Amazon SQS

Para publicar notificações em um tópico do SNS ou em uma fila do SQS, primeiro conceda permissões ao Amazon S3. Para conceder essas permissões, anexe uma política do AWS Identity and Access Management (IAM) ao tópico do SNS ou à fila do SQS de destino. Para saber mais sobre as políticas do IAM necessárias, consulte [Conceder permissões para publicar mensagens em um tópico do SNS ou em uma fila do SQS](#).

Para obter informações sobre a integração do SNS e do SQS com o X-Ray, consulte [Amazon SNS e AWS X-Ray](#) e [Amazon SQS e AWS X-Ray](#).

AWS Lambda

Ao usar o console do Amazon S3 para configurar notificações de eventos em um bucket do S3 para uma função do Lambda, o console configurará as permissões necessárias na função do Lambda para que o Amazon S3 tenha as permissões para invocar a função no bucket. Para obter mais informações, consulte [How Do I Enable and Configure Event Notifications for an S3 Bucket?](#) no Guia do usuário do Amazon Simple Storage Service.

Você também pode conceder permissões do Amazon S3 do AWS Lambda para invocar a função do Lambda. Para obter mais informações, consulte [Tutorial: Using AWS Lambda with Amazon S3](#) no Guia do desenvolvedor do AWS Lambda.

Para obter mais informações sobre a integração do Lambda com o X-Ray, consulte [Instrumentação do código Java no AWS Lambda](#).

AWS Distro para OpenTelemetry e AWS X-Ray

Use o AWS Distro para OpenTelemetry (ADOT) para coletar e enviar métricas e rastreamentos ao AWS X-Ray e outras soluções de monitoramento, como o Amazon CloudWatch, o Amazon OpenSearch Service e o Amazon Managed Service for Prometheus.

AWS Distro para OpenTelemetry

O AWS Distro para OpenTelemetry (ADOT) é uma distribuição da AWS baseada no projeto OpenTelemetry da Cloud Native Computing Foundation (CNCF). O OpenTelemetry fornece um único conjunto de APIs, bibliotecas e agentes de código aberto para coletar métricas e rastreamentos distribuídos. Esse kit de ferramentas é uma distribuição de componentes precedentes do OpenTelemetry, como SDKs, agentes de instrumentação automática e coletores compatíveis com a AWS e testados, otimizados e protegidos por ela.

Com o ADOT, os engenheiros podem instrumentar as aplicações uma vez e enviar métricas e rastreamentos correlacionados para várias soluções de monitoramento da AWS, como o Amazon CloudWatch, o AWS X-Ray, o Amazon OpenSearch Service e o Amazon Managed Service for Prometheus.

O ADOT é integrado a um número crescente de Serviços da AWS para simplificar o envio de rastreamentos e métricas para soluções de monitoramento, como o X-Ray. Alguns exemplos de serviços integrados ao ADOT incluem:

- **AWS Lambda:** as camadas do AWS Lambda gerenciadas para o ADOT fornecem uma experiência de usuário plug-and-play instrumentando automaticamente uma função do Lambda e empacotando o OpenTelemetry com uma configuração pronta para uso para o AWS Lambda e o X-Ray em uma camada fácil de configurar. Os usuários podem habilitar e desabilitar o OpenTelemetry para a função do Lambda sem alterar o código. Para obter mais informações, consulte [AWS Distro for OpenTelemetry Lambda](#)
- **Amazon Elastic Container Service (ECS):** colete métricas e rastreamentos de aplicações do Amazon ECS usando o coletor AWS Distro para OpenTelemetry para enviá-los ao X-Ray e a outras soluções de monitoramento. Para obter mais informações, consulte [Coleta de dados de rastreamento de aplicações](#) no Guia do desenvolvedor do Amazon ECS.
- **AWS App Runner:** o App Runner comporta o envio de rastreamentos ao X-Ray usando o AWS Distro para OpenTelemetry (ADOT). Use os SDKs do ADOT para coletar dados de rastreamento para aplicações containerizadas e use o X-Ray para analisar e obter informações sobre a aplicação instrumentada. Para obter mais informações, consulte [AWS App Runner e X-Ray](#).

Para obter mais informações sobre o AWS Distro para OpenTelemetry, bem como sobre a integração com outros Serviços da AWS, consulte a [documentação do AWS Distro para OpenTelemetry](#).

Para obter mais informações sobre a instrumentação da aplicação com o AWS Distro para OpenTelemetry e o X-Ray, consulte [Instrumenting your application with the AWS Distro for OpenTelemetry](#).

Rastrear as alterações da configuração de criptografia do X-Ray com o AWS Config

O AWS X-Ray se integra ao AWS Config para registrar as alterações de configuração feitas nos recursos de criptografia do X-Ray. Você pode usar o AWS Config para inventariar os recursos de criptografia do X-Ray, auditar o histórico de configuração do X-Ray e enviar notificações com base nas alterações de recursos.

O AWS Config é compatível com o registro em log para as seguintes alterações de recursos de criptografia do X-Ray como eventos:

- Alterações de configuração: alteração ou adição de uma chave de criptografia ou reversão para a configuração de criptografia padrão do X-Ray.

Use as instruções a seguir para saber como criar uma conexão básica entre o X-Ray e o AWS Config.

Criar um acionador de função do Lambda

Você precisa ter o ARN de uma função personalizada do AWS Lambda para que possa gerar uma regra personalizada do AWS Config. Siga estas instruções para criar uma função básica com Node.js que retorne um valor, compatível ou não, para o AWS Config com base no estado do recurso `XrayEncryptionConfig`.

Para criar uma função Lambda com um trigger de alteração de AWS:: XrayEncryptionConfig

1. Abra o [console do lambda](#). Escolha a opção Criar função.
2. Selecione Blueprints (Esquemas) e, em seguida, filtre a biblioteca de esquemas para o esquema `config-rule-change-triggered`. Clique no link do nome do esquema ou selecione Configure (Configurar) para continuar.

3. Defina os seguintes campos para configurar o esquema:
 - Em Nome, digite um nome.
 - Para Role, selecione Create new role from template(s).
 - Para Role name, digite um nome.
 - Em Policy templates (Modelos de política), selecione AWS Config Rules permissions (Permissões de regras do &CC;).
4. Selecione Create function (Criar função) para criar e exibir sua função no console do AWS Lambda.
5. Edite o código da função para substituir `AWS::EC2::Instance` por `AWS::XrayEncryptionConfig`. Você também pode atualizar o campo de descrição para refletir essa alteração.

Código padrão

```
if (configurationItem.resourceType !== 'AWS::EC2::Instance') {
    return 'NOT_APPLICABLE';
} else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
    return 'COMPLIANT';
}
return 'NON_COMPLIANT';
```

Código atualizado

```
if (configurationItem.resourceType !== 'AWS::XRay::EncryptionConfig') {
    return 'NOT_APPLICABLE';
} else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
    return 'COMPLIANT';
}
return 'NON_COMPLIANT';
```

6. Adicione o seguinte ao seu perfil de execução no IAM para acesso ao X-Ray. Essas permissões autorizam o acesso somente leitura aos recursos do X-Ray. Uma falha ao fornecer acesso aos recursos adequados resultará em uma mensagem fora do escopo do AWS Config quando ele avaliar a função do Lambda associada à regra.

```
{
```

```
"Sid": "Stmt1529350291539",
"Action": [
    "xray:GetEncryptionConfig"
],
"Effect": "Allow",
"Resource": "*"
}
```

Criar uma regra do AWS Config personalizada para o X-Ray

Quando a função do Lambda for criada, anote o respectivo ARN e acesse o console do AWS Config para criar uma regra personalizada.

Como criar uma regra do AWS Config para o X-Ray

1. Abra a página [Regras de AWS Config do console](#).
2. Selecione Add rule (Adicionar regra)e, em seguida, Add custom rule (Adicionar regra personalizada).
3. Em ARN da função do AWS Lambda, insira o ARN associado à função do Lambda que você deseja usar.
4. Escolha o tipo de trigger a ser definido:
 - Alterações de configuração: o AWS Config aciona a avaliação quando algum recurso que corresponde ao escopo da regra tem sua configuração alterada. A avaliação é executada depois que o AWS Config envia uma notificação de alteração de item de configuração.
 - Periódico: AWS Config executa avaliações para a regra em uma frequência definida por você (por exemplo, a cada 24 horas).
5. Em Tipo de recurso, escolha EncryptionConfig na seção do X-Ray.
6. Selecione Salvar.

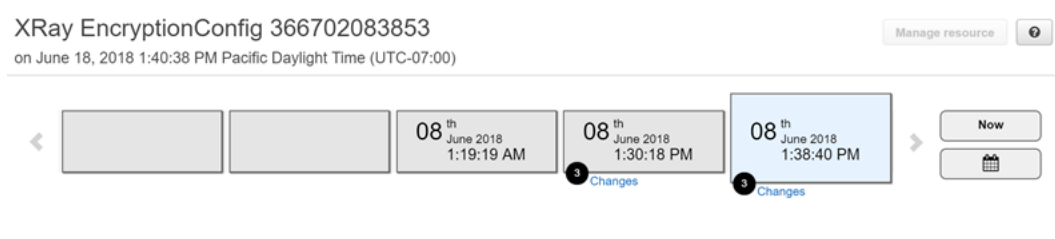
O console do AWS Config começará a avaliar a conformidade da regra imediatamente. A avaliação pode demorar alguns minutos para ser concluída.

Agora que essa regra é compatível, o AWS Config pode começar a compilar um histórico de auditoria. O AWS Config registra as alterações de recursos na forma de um cronograma. Para cada alteração na linha do tempo de eventos, o AWS Config gera uma tabela em um formato de/para a fim

de mostrar o que foi alterado na representação JSON da chave de criptografia. As duas alterações de campo associadas a EncryptionConfig são Configuration.type e Configuration.keyID.

Resultados de exemplo

Veja a seguir um exemplo de uma linha do tempo do AWS Config mostrando as alterações feitas em datas e horários específicos.



Veja a seguir um exemplo de uma entrada de alteração do AWS Config. O formato de/para ilustra o que foi alterado. Este exemplo mostra que as configurações de criptografia padrão do X-Ray foram alteradas para uma chave de criptografia definida.



Notificações do Amazon SNS

Para receber notificação sobre alterações de configuração, configure o AWS Config para publicar notificações do Amazon SNS. Para mais informações, consulte [Monitoramento de AWS Config mudanças de recursos por e-mail](#).

AWS AppSync e AWS X-Ray

Você pode habilitar e rastrear solicitações para o AWS AppSync. Para obter mais informações, consulte [Rastreamento com o AWS X-Ray](#) para obter instruções.

Quando o rastreamento do X-Ray é habilitado para uma API do AWS AppSync, um [perfil vinculado ao serviço](#) do AWS Identity and Access Management é criado automaticamente em sua conta com

as permissões apropriadas. Isso permite que o AWS AppSync envie rastreamentos para o X-Ray de forma segura.

Compatibilidade com o rastreamento ativo do Amazon API Gateway para o AWS X-Ray

Você pode usar o X-Ray para rastrear e analisar solicitações do usuário à medida que elas passam pelas APIs do Amazon API Gateway em direção aos serviços subjacentes. O API Gateway comporta o rastreamento do X-Ray para todos os tipos de endpoint do API Gateway: regional, otimizado para borda e privado. Você pode usar o X-Ray com o Amazon API Gateway em todas as Regiões da AWS em que o X-Ray está disponível. Para obter mais informações, consulte [Trace API Gateway API Execution with AWS X-Ray](#), no Guia do desenvolvedor do Amazon API Gateway.

Note

O X-Ray só comporta o rastreamento de APIs REST por meio do API Gateway.

O Amazon API Gateway oferece compatibilidade com o [rastreamento ativo](#) para o AWS X-Ray. Habilite o rastreamento ativo em seus estágios de API para realizar amostragens das solicitações de entrada e enviar rastreamentos ao X-Ray.

Para habilitar o rastreamento ativo em um estágio de API

1. Abra o console do API Gateway em <https://console.aws.amazon.com/apigateway/>.
2. Escolha uma API.
3. Escolha um estágio.
4. Na guia Logs/Rastreamento, escolha Habilitar rastreamento com X-Ray e selecione Salvar alterações.
5. Escolha Resources (Recursos) no painel de navegação à esquerda.
6. Para reimplantar a API com as novas configurações, escolha o menu suspenso Ações e selecione Implantar API.

O API Gateway usa as regras de amostragem que você define no console do X-Ray para determinar quais solicitações serão registradas. Você pode criar regras que se apliquem apenas às APIs ou

que se apliquem somente às solicitações que contenham determinados cabeçalhos. O API Gateway registra cabeçalhos em atributos no segmento, bem como detalhes sobre o estágio e a solicitação. Para obter mais informações, consulte [Configurar regras de amostragem](#).

Note

Ao rastrear APIs REST com a [integração HTTP](#) do API Gateway, o nome do serviço de cada segmento é definido como o caminho do URL da solicitação desde o API Gateway até seu endpoint de integração HTTP, o que resulta em um nó de serviço no mapa de serviço do X-Ray para cada caminho de URL exclusivo. Um grande número de caminhos de URL pode fazer com que o mapa de serviço exceda o limite de dez mil nós, resultando em um erro. Para minimizar o número de nós de serviço criados pelo API Gateway, considere a possibilidade de passar parâmetros na string de consulta do URL ou no corpo da solicitação via POST. Qualquer abordagem garantirá que os parâmetros não façam parte do caminho do URL, o que pode resultar em menos caminhos de URL e nós de serviço distintos.

Para todas as solicitações de entrada, o Gateway da API adiciona um [cabeçalho de rastreamento](#) às solicitações HTTP de entrada que ainda não têm um.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Formato do ID de rastreamento do X-Ray

Um `trace_id` do X-Ray consiste em três números separados por hifens. Por exemplo, `1-58406520-a006649127e371903a2de979`. Isso inclui:

- O número da versão, que é 1.
- A hora da solicitação original, em horário epoch Unix, com 8 dígitos hexadecimais.

Por exemplo, 10h no dia 1º de dezembro de 2016 PST equivale a 1480615200 segundos em horário epoch ou a 58406520 em dígitos hexadecimais.

- Um identificador globalmente exclusivo de 96 bits para o rastreamento com 24 dígitos hexadecimais.

Se o rastreamento ativo estiver desabilitado, o estágio ainda registrará um segmento se a solicitação vier de um serviço que realizou a amostragem da solicitação e iniciou um rastreamento. Por

exemplo, uma aplicação web instrumentada pode chamar uma API do API Gateway com um cliente HTTP. Quando você instrumenta um cliente HTTP com o X-Ray SDK, ele adiciona um cabeçalho de rastreamento à solicitação de saída que contém a decisão de amostragem. O API Gateway lê o cabeçalho de rastreamento e cria um segmento para amostras de solicitações.

Se você usar o API Gateway para [gerar um SDK do Java para a API](#), poderá instrumentar o cliente do SDK adicionando um manipulador de solicitações ao compilador de clientes, da mesma forma que instrumentaria manualmente um cliente do SDK da AWS. Para obter instruções, consulte [Rastreamento chamadas AWS do SDK com o X-Ray SDK for Java](#).

Amazon EC2 e AWS App Mesh

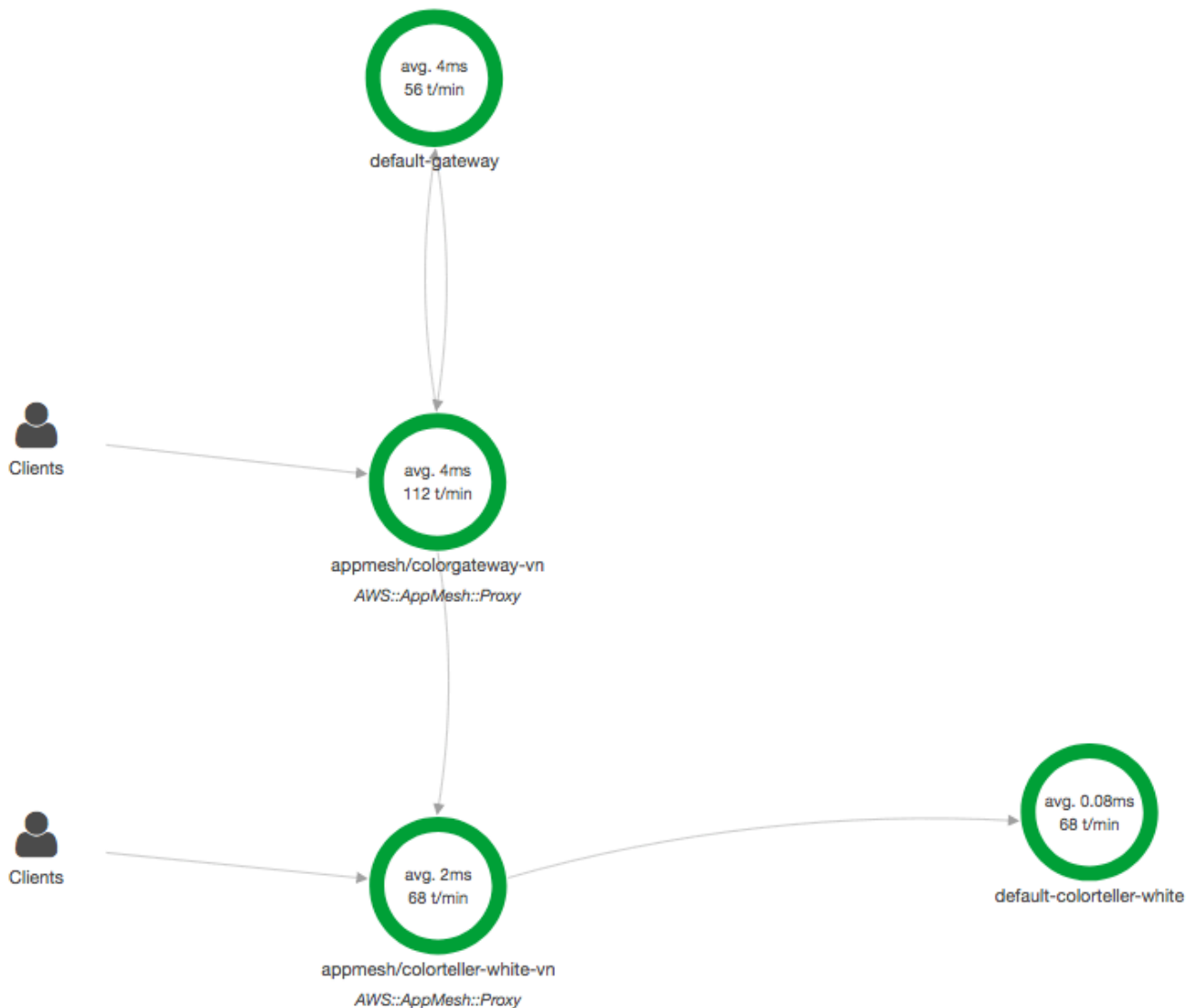
O AWS X-Ray integra-se ao [AWS App Mesh](#) para gerenciar proxies Envoy para microsserviços. O App Mesh fornece uma versão do Envoy que você pode configurar para enviar dados de rastreamento ao daemon X-Ray executado em um contêiner da mesma tarefa ou pod. O X-Ray oferece suporte a rastreamento com os seguintes serviços compatíveis com o App Mesh:

- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Amazon Elastic Compute Cloud (Amazon EC2)

Use as instruções a seguir para saber como habilitar o rastreamento do X-Ray por meio do App Mesh.

Service map

Enter a service name to find and select the node on map



Para configurar o proxy do Envoy para enviar dados ao X-Ray, defina `ENABLE_ENVOY_XRAY_TRACING` como [variável de ambiente](#) em sua definição de contêiner.

Note

No momento, a versão App Mesh do Envoy não envia rastreamentos com base nas [regras de amostragem](#) configuradas. Em vez disso, ele usa uma taxa de amostragem fixa de 5%

para o Envoy versão 1.16.3 ou mais recente ou uma taxa de amostragem de 50% para versões do Envoy anteriores à 1.16.3.

Exemplo Definição de contêiner do Envoy para o Amazon ECS

```
{
  "name": "envoy",
  "image": "public.ecr.aws/appmesh/aws-appmesh-envoy:envoy-version",
  "essential": true,
  "environment": [
    {
      "name": "APPMESH_VIRTUAL_NODE_NAME",
      "value": "mesh/myMesh/virtualNode/myNode"
    },
    {
      "name": "ENABLE_ENVOY_XRAY_TRACING",
      "value": "1"
    }
  ],
  "healthCheck": {
    "command": [
      "CMD-SHELL",
      "curl -s http://localhost:9901/server_info | cut -d' ' -f3 | grep -q live"
    ],
    "startPeriod": 10,
    "interval": 5,
    "timeout": 2,
    "retries": 3
  }
}
```

Note

Para saber mais sobre os endereços de regiões do Envoy disponíveis, consulte [Imagem do Envoy](#) no Guia do usuário do AWS App Mesh.

Para obter detalhes sobre como executar o daemon do X-Ray em um contêiner, consulte [Executar o daemon do X-Ray no Amazon ECS](#). Para ver uma aplicação de exemplo que inclui uma malha de serviços, um microsserviço, o proxy do Envoy e o daemon do X-Ray, implante o exemplo colorapp no [Repositório GitHub de exemplos do App Mesh](#).

Saiba mais

- [Conceitos básicos do AWS App Mesh](#)
- [Conceitos básicos do AWS App Mesh e do Amazon ECS](#)

AWS App Runner e X-Ray

O AWS App Runner é um AWS service (Serviço da AWS) que oferece uma maneira rápida, simples e econômica de implantar diretamente em uma aplicação web escalável e segura na Nuvem AWS usando o código-fonte ou uma imagem de contêiner. Você não precisa aprender novas tecnologias, decidir qual serviço de computação usar nem saber como provisionar e configurar os recursos da AWS. Para obter mais informações, consulte [What is AWS App Runner?](#).

O AWS App Runner envia rastreamentos ao X-Ray por meio da integração com o [AWS Distro para OpenTelemetry](#) (ADOT). Use os SDKs do ADOT para coletar dados de rastreamento para aplicações containerizadas e use o X-Ray para analisar e obter informações sobre a aplicação instrumentada. Para obter mais informações, consulte [Tracing for your App Runner application with X-Ray](#).

Registro em log de chamadas de API do X-Ray com o AWS CloudTrail

O AWS X-Ray é integrado ao [AWS CloudTrail](#), um serviço que fornece um registro das ações realizadas por um usuário, um perfil ou um AWS service (Serviço da AWS). O CloudTrail captura todas as chamadas API para X-Ray como eventos. As chamadas capturadas incluem as chamadas do console do X-Ray e as chamadas de código para as operações da API do X-Ray. Ao fazer uso das informações coletadas pelo CloudTrail, é possível determinar a solicitação feita ao X-Ray, o endereço IP no qual a solicitação foi feita, quando a solicitação foi feita e detalhes adicionais.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário raiz ou credenciais de usuário.
- Se a solicitação foi feita em nome de um usuário do Centro de Identidade do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

Quando a Conta da AWS é criada, o CloudTrail torna-se ativo nela e você tem acesso automático ao Histórico de eventos do CloudTrail. O Histórico de eventos do CloudTrail fornece um registro visualizável, pesquisável, baixável e imutável dos últimos 90 dias de eventos de gerenciamento gravados em uma Região da AWS. Para obter mais informações, consulte [Trabalhar com histórico de eventos do CloudTrail](#) no Guia do usuário do AWS CloudTrail. Não há cobranças do CloudTrail pela visualização do Histórico de eventos.

Para obter um registro contínuo de eventos em sua Conta da AWS nos últimos 90 dias, crie uma trilha ou um armazenamento de dados de eventos do [CloudTrail Lake](#).

Trilhas do CloudTrail

Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. As trilhas criadas usando o Console de gerenciamento da AWS são de várias regiões. Só é possível criar uma trilha de região única ou de várias regiões usando a AWS CLI. Criar uma trilha de várias regiões é uma prática recomendada, pois você captura atividades em todas as Regiões da AWS da conta. Ao criar uma trilha de região única, é possível visualizar somente os eventos registrados na Região da AWS da trilha. Para obter mais informações sobre trilhas, consulte [Criar uma trilha para a Conta da AWS](#) e [Criar uma trilha para uma organização](#) no Guia do usuário do AWS CloudTrail.

Uma cópia dos eventos de gerenciamento em andamento pode ser entregue no bucket do Amazon S3 sem nenhum custo via CloudTrail com a criação de uma trilha; no entanto, há cobranças de armazenamento do Amazon S3. Para obter mais informações sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#). Para receber informações sobre a definição de preços do Amazon S3, consulte [Definição de preços do Amazon S3](#).

Armazenamentos de dados de eventos do CloudTrail Lake

O CloudTrail Lake permite executar consultas baseadas em SQL nos eventos. O CloudTrail Lake converte eventos existentes em formato JSON baseado em linhas para o formato [Apache ORC](#). O ORC é um formato colunar de armazenamento otimizado para recuperação rápida de dados. Os eventos são agregados em armazenamentos de dados de eventos, que são coleções imutáveis de eventos baseados nos critérios selecionados com a aplicação de [seletores de eventos avançados](#). Os seletores que aplicados a um armazenamento de dados de eventos controlam quais eventos persistem e estão disponíveis para consulta. Para obter mais informações sobre o CloudTrail Lake, consulte [Trabalhar com o AWS CloudTrail Lake](#), no Guia do usuário do AWS CloudTrail.

Os armazenamentos de dados de eventos e consultas do CloudTrail Lake incorrem em custos. Ao criar um armazenamento de dados de eventos, você escolhe a [opção de preço](#) que deseja usar para ele. A opção de preço determina o custo para a ingestão e para o armazenamento de eventos, e o período de retenção padrão e máximo para o armazenamento de dados de eventos. Para obter mais informações sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#).

Tópicos

- [Eventos de gerenciamento no CloudTrail](#)
- [Eventos de dados no CloudTrail](#)
- [Exemplos de eventos X-Ray](#)

Eventos de gerenciamento no CloudTrail

O AWS X-Ray se integra ao AWS CloudTrail para registrar ações de API realizadas por um usuário, uma função ou um AWS service (Serviço da AWS) no X-Ray. Você pode usar o CloudTrail para monitorar solicitações de API do X-Ray em tempo real e armazenar logs no Amazon S3, Amazon CloudWatch Logs e Amazon CloudWatch Events. O X-Ray é compatível com as seguintes ações como eventos nos arquivos de log do CloudTrail:

Ações de API compatíveis

- [PutEncryptionConfig](#)
- [GetEncryptionConfig](#)
- [CreateGroup](#)
- [UpdateGroup](#)
- [DeleteGroup](#)
- [GetGroup](#)
- [GetGroups](#)
- [GetInsight](#)
- [GetInsightEvents](#)
- [GetInsightImpactGraph](#)
- [GetInsightSummaries](#)
- [GetSamplingStatisticSummaries](#)

Eventos de dados no CloudTrail

[Os eventos de dados](#) fornecem informações sobre as operações de recursos realizadas em ou em um recurso (por exemplo [PutTraceSegments](#), que carrega documentos do segmento para o X-Ray).

Também são conhecidas como operações de plano de dados. Eventos de dados geralmente são atividades de alto volume. Por padrão, o CloudTrail não registra eventos de dados em log. O Histórico de eventos do CloudTrail não registra eventos de dados.

Há cobranças adicionais para eventos de dados. Para obter mais informações sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#).

É possível registrar em log eventos de dados para os tipos de recurso do X-Ray usando o console do CloudTrail, a AWS CLI ou as operações de API do CloudTrail. Para obter mais informações sobre como registrar eventos de dados em log, consulte [Registrar eventos de dados com o Console de gerenciamento da AWS](#) e [Registrar eventos de dados com a AWS Command Line Interface](#) no Guia do usuário do AWS CloudTrail.

A tabela a seguir lista o tipo de recurso do Amazon S3 para o qual você pode registrar eventos de dados em log. A coluna Tipo de evento de dados (console) mostra o valor a ser escolhido na lista Tipo de evento de dados no console do CloudTrail. A coluna do valor `resources.type` mostra o valor de `resources.type` que você especificaria ao configurar seletores de eventos avançados usando a AWS CLI ou as APIs do CloudTrail. A coluna APIs de dados registradas no CloudTrail mostra as chamadas de API registradas no CloudTrail para o tipo de recurso.

Tipo de evento de dados (console)	valor <code>resources.type</code>	APIs de dados registradas no CloudTrail
Rastreamentos do X-Ray	<code>AWS::XRay::Trace</code>	• PutTraceSegments • GetTraceSummaries • GetTraceGraph • GetServiceGraph • BatchGetTraces • GetTimeSeriesServiceStatistics • PutTelemetryRecords • GetSamplingTargets

É possível configurar seletores de eventos avançados para filtrar os campos `eventName` e `readOnly` para registrar em log somente os eventos que são importantes para você. No entanto, você não pode selecionar eventos adicionando o seletor de `resources.ARN` campo, porque os rastreamentos de X-Ray não têm ARNs. Para obter mais informações sobre esses campos, consulte [AdvancedFieldSelector](#) na Referência de API do AWS CloudTrail. Veja a seguir um exemplo de como executar o [put-event-selectors](#) AWS CLI comando para registrar eventos de dados em uma trilha do CloudTrail. Você deve executar o comando ou especificar a região na qual a trilha foi criada; caso contrário, a operação retornará uma `InvalidHomeRegionException` exceção.

```
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"
    }
  ]
}'
```

Exemplos de eventos X-Ray

Exemplos de eventos de gerenciamento

O exemplo a seguir mostra uma entrada de log do CloudTrail para a ação `GetEncryptionConfig`.

Example

```
{
  "eventVersion"=>"1.05",
  "userIdentity"=>{
    "type"=>"AssumedRole",
    "principalId"=>"AR0AJVHBZWD3DN6CI2MHM: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"=>"AR0AJVHBZWD3DN6CI2MHH",
        "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"
}

```

Exemplos de dados de eventos

Veja a seguir um exemplo da entrada do log de eventos de PutTraceSegments dados do X-Ray no CloudTrail.

Example

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AR0AWYXPW54Y4NEXAMPLE: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"  
}
```

Integração ao CloudWatch para o X-Ray

AWS X-Ray se integra ao [CloudWatch Application Signals](#), CloudWatch RUM e CloudWatch Synthetics para facilitar o monitoramento da integridade das aplicações. Habilite seu aplicativo para o Application Signals para monitorar e solucionar problemas da integridade operacional dos seus serviços, páginas de clientes, canários do Synthetics e dependências de serviços.

O ServiceLens fornece uma visão completa de seus serviços correlacionando métricas, logs e rastreamentos, para ajudá-lo a identificar rapidamente gargalos de desempenho e identificar usuários atingidos.

Com o CloudWatch RUM, você pode realizar o monitoramento real do usuário para coletar e visualizar dados do lado do cliente sobre a performance da aplicação web com base em sessões reais do usuário praticamente em tempo real. Com o AWS X-Ray e o CloudWatch RUM, você pode analisar e depurar o caminho da solicitação a partir dos usuários finais da aplicação até os serviços subsequentes gerenciados pela AWS. Isso ajuda a identificar tendências e erros de latência que afetam os usuários finais.

Tópicos

- [CloudWatch RUM e AWS X-Ray](#)
- [Depurar canários sintéticos do CloudWatch usando o X-Ray](#)

CloudWatch RUM e AWS X-Ray

Com o Amazon CloudWatch RUM, você pode realizar o monitoramento real do usuário para coletar e visualizar dados do lado do cliente sobre a performance da aplicação web com base em sessões reais do usuário praticamente em tempo real. Com o AWS X-Ray e o CloudWatch RUM, você pode analisar e depurar o caminho da solicitação a partir dos usuários finais da aplicação até os serviços subsequentes gerenciados pela AWS. Isso ajuda a identificar tendências e erros de latência que afetam os usuários finais.

Depois que você ativa o rastreamento do X-Ray das sessões do usuário, o CloudWatch RUM adiciona um cabeçalho de rastreamento do X-Ray às solicitações HTTP permitidas e registra um segmento do X-Ray para solicitações HTTP permitidas. Você pode visualizar rastreamentos e segmentos dessas sessões de usuário no console do X-Ray e do CloudWatch, incluindo o mapa de serviço do X-Ray.

Note

O CloudWatch RUM não se integra a regras de amostragem do X-Ray. Em vez disso, escolha uma porcentagem de amostragem ao configurar a aplicação para usar o CloudWatch RUM. Os rastreamentos enviados do CloudWatch RUM podem incorrer em custos adicionais. Para obter mais informações, consulte [Preços do AWS X-Ray](#).

Por padrão, os rastreamentos do lado do cliente enviados do CloudWatch RUM não são conectados a rastreamentos do lado do servidor. Para conectar rastreamentos do lado do cliente com rastreamentos do lado do servidor, configure o cliente da web do CloudWatch RUM para adicionar um cabeçalho de rastreamento do X-Ray a essas solicitações HTTP.

Warning

Configurar o cliente da web do CloudWatch RUM para adicionar um cabeçalho de rastreamento do X-Ray a solicitações HTTP pode fazer com que o compartilhamento de recursos de origem cruzada (CORS) falhe. Para evitar isso, adicione o cabeçalho HTTP `X-Amzn-Trace-Id` à lista de cabeçalhos permitidos na configuração do CORS do seu serviço subsequente. Se você estiver usando o API Gateway como serviço subsequente, consulte [Habilitar o CORS para um recurso da API REST](#). É altamente recomendável que você teste sua aplicação antes de adicionar um cabeçalho de rastreamento do X-Ray do lado do cliente em um ambiente de produção. Para obter mais informações, consulte a [Documentação do cliente da Web do CloudWatch RUM](#).

Para obter mais informações sobre o monitoramento real de usuários no CloudWatch, consulte [Usar o CloudWatch RUM](#). Para configurar a aplicação para usar o CloudWatch RUM, incluindo o rastreamento de sessões do usuário com o X-Ray, consulte [Configurar uma aplicação para usar o CloudWatch RUM](#).

Depurar canários sintéticos do CloudWatch usando o X-Ray

O CloudWatch Synthetics é um serviço totalmente gerenciado que permite monitorar endpoints e APIs usando canários com script que funcionam 24 horas por dia, uma vez por minuto.

Você pode personalizar scripts canários para verificar se há alterações em:

- Disponibilidade
- Latência
- Transações
- Links quebrados ou inoperantes
- Conclusões de tarefas passo a passo
- Erros de carregamento de página
- Carregar latências para ativos de interface do usuário
- Fluxos complexos do assistente
- Fluxos de checkout em seu aplicativo

Os canários seguem as mesmas rotas e executam as mesmas ações e comportamentos que seus clientes, e verificam a experiência do cliente continuamente.

Para saber mais sobre como configurar testes do Synthetics, consulte [Using Synthetics to Create and Manage Canaries](#).



Os exemplos a seguir mostram casos de uso em comum para problemas de depuração que seus canários Synthetics levantam. Cada exemplo demonstra uma estratégia chave para depuração usando o mapa de serviço ou o console do X-Ray Analytics.

Para obter mais informações sobre como ler e interagir com o mapa de serviço, consulte [Exibição do mapa de serviços](#).

Para obter mais informações sobre como ler e interagir com o console do X-Ray Analytics, consulte [Interacting with the AWS X-Ray Analytics Console](#).

Tópicos

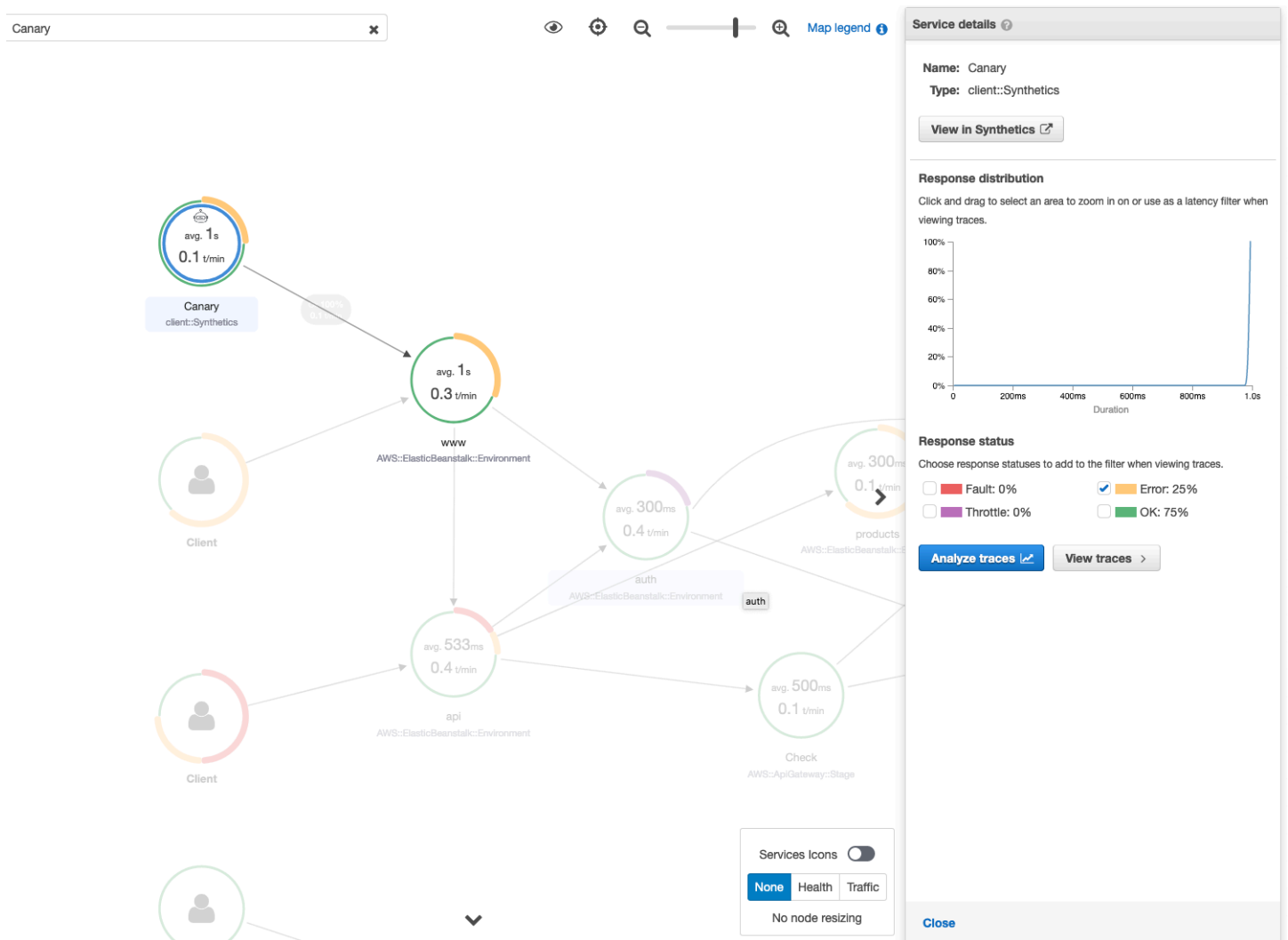
- [Exibir canários com relatórios de erros aumentados no mapa de serviços](#)

- [Usar mapas de rastreamento em rastreamentos individuais para exibir cada solicitação detalhadamente](#)
- [Determinar a causa raiz de falhas contínuas nos serviços upstream e downstream](#)
- [Identificar gargalos e tendências de desempenho](#)
- [Comparar taxas de latência e erro ou falha antes e depois das alterações](#)
- [Determinar a cobertura canária necessária para todas as APIs e URLs](#)
- [Usar grupos para se concentrar em testes do Synthetics](#)

Exibir canários com relatórios de erros aumentados no mapa de serviços

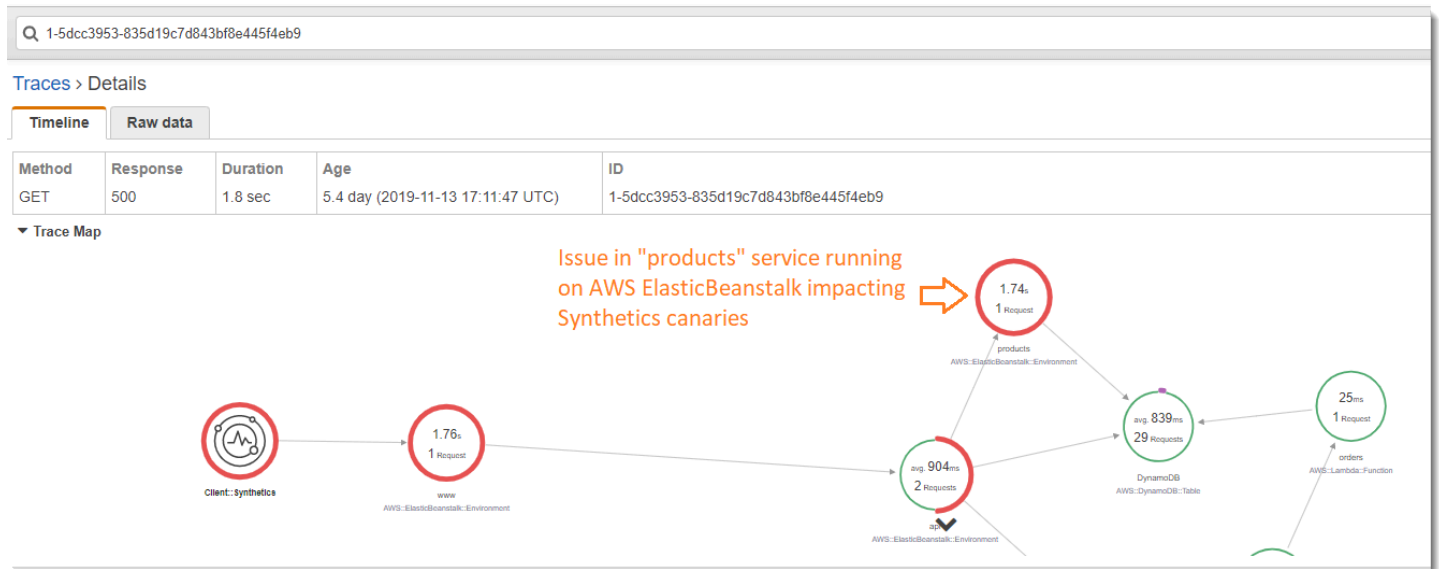
Para ver quais canários têm aumento de erros, falhas, taxas de controle de utilização ou tempos de resposta lentos dentro do mapa de serviço do X-Ray, é possível realçar nós de cliente do canário do Synthetics usando o [filtro](#) `Client::Synthetic`. Clicar em um nó exibe a distribuição do tempo de resposta de toda a solicitação. Clicar em uma borda entre dois nós mostra detalhes sobre as solicitações que percorreram aquela conexão. Você também pode visualizar nós inferidos “remotos” para serviços subsequentes relacionados no mapa de serviço.

Quando você clica no nó Synthetics, há um botão Visualizar em Synthetics no painel lateral que redireciona você para o console do Synthetics, onde você pode verificar os detalhes do canário.



Usar mapas de rastreamento em rastreamentos individuais para exibir cada solicitação detalhadamente

Para determinar qual serviço resulta na maior latência ou está causando um erro, chame o mapa de rastreamento selecionando o rastreamento no mapa de serviço. Mapas de rastreamento individuais exibem o caminho de ponta a ponta de uma única solicitação. Use isso para entender os serviços invocados e visualizar os serviços upstream e downstream.



Determinar a causa raiz de falhas contínuas nos serviços upstream e downstream

Depois de receber um alarme do CloudWatch por falhas em um canário do Synthetics, use a modelagem estatística nos dados de rastreamento no X-Ray para determinar a provável causa raiz do problema no console do X-Ray Analytics. No console do Analytics, a tabela Causa raiz do tempo de resposta mostra os caminhos de entidades registrados. O X-Ray determina qual caminho no rastreamento é a causa mais provável do tempo de resposta. O formato indica uma hierarquia de entidades encontradas, terminando em uma causa raiz do tempo de resposta.

O exemplo a seguir mostra que o teste do Synthetics para a API “XXX” em execução no API Gateway está falhando devido a uma exceção de capacidade de throughput da tabela do Amazon DynamoDB.



AWS:CANARY_ARN	COUNT
arn:aws:synthetics:us-east-1:779168132807:canary:www-test	118

Annotation.aws.canary_arn

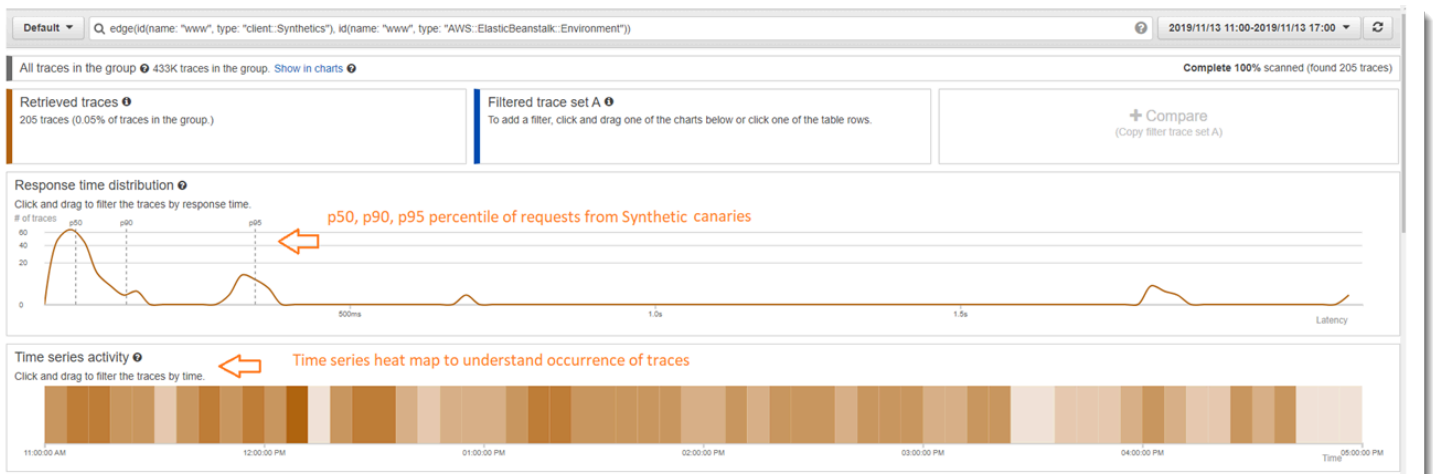
Annotation.cold_start

Annotation.credentials_cached

Annotation.queries

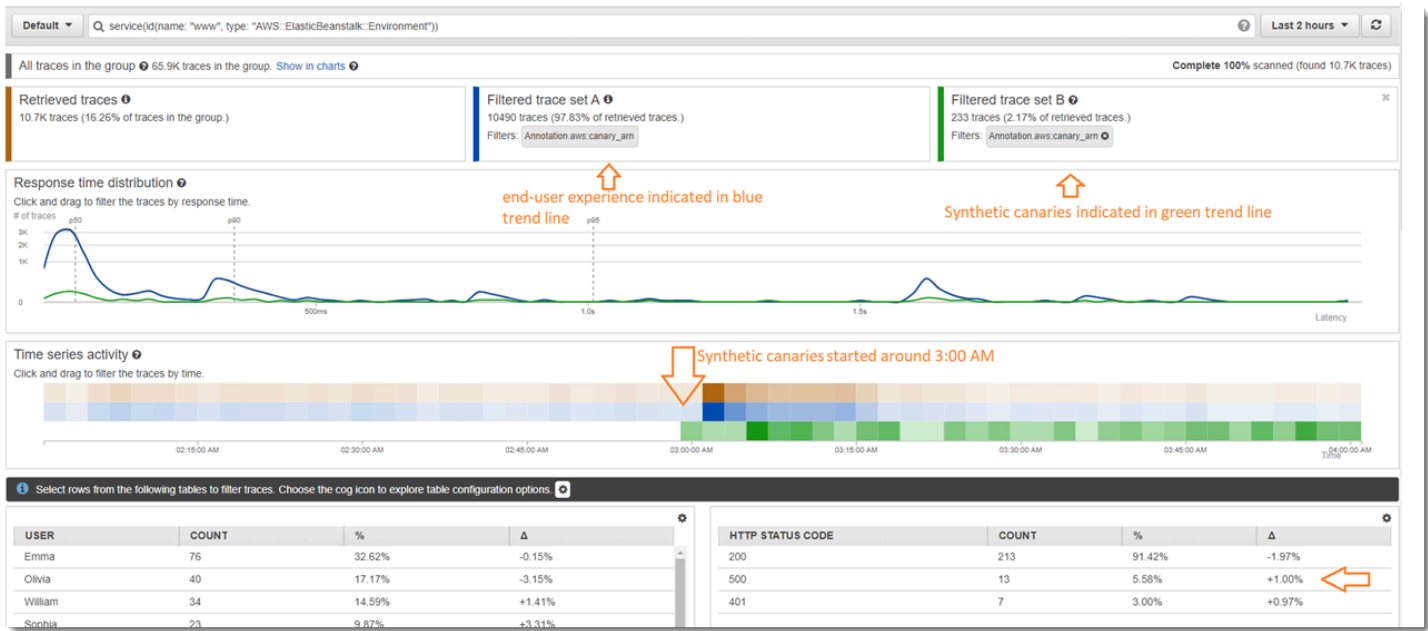
Identificar gargalos e tendências de desempenho

Você pode visualizar tendências no desempenho de seu endpoint ao longo do tempo usando o tráfego contínuo de seus canários Synthetic para preencher um mapa de rastreamento durante um período de tempo.



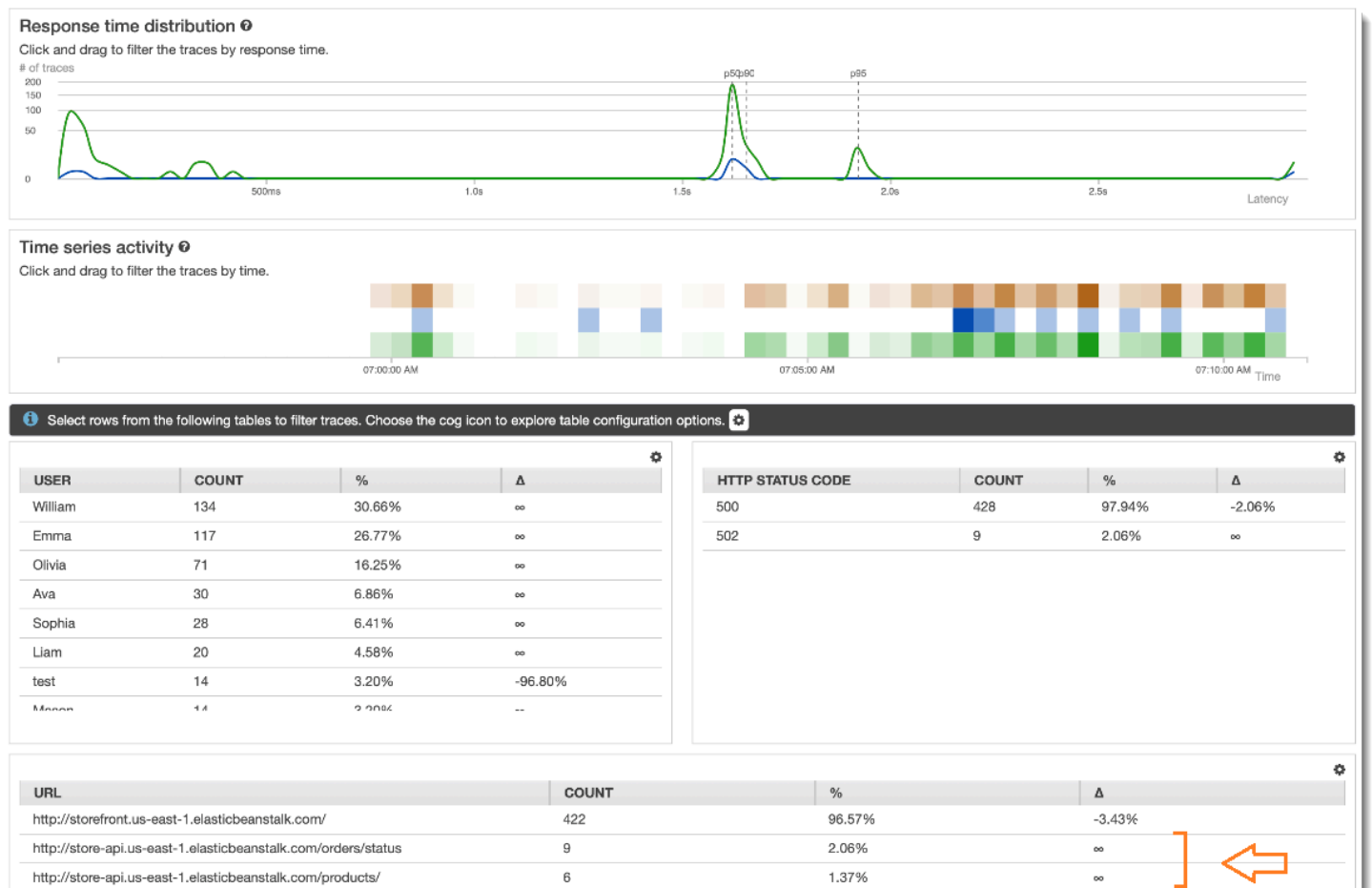
Comparar taxas de latência e erro ou falha antes e depois das alterações

Identifique a hora em que ocorreu uma alteração para correlacionar essa alteração com um aumento de problemas capturados por seus canários. Use o console do X-Ray Analytics para definir os intervalos de tempo anteriores e posteriores como diferentes conjuntos de rastreamento, criando uma diferenciação visual na distribuição do tempo de resposta.



Determinar a cobertura canária necessária para todas as APIs e URLs

Use o X-Ray Analytics para comparar a experiência de canários com os usuários. A interface do usuário abaixo mostra uma linha de tendência azul para canários e uma linha verde para os usuários. Também é possível identificar que dois dos três URLs não têm testes canários.

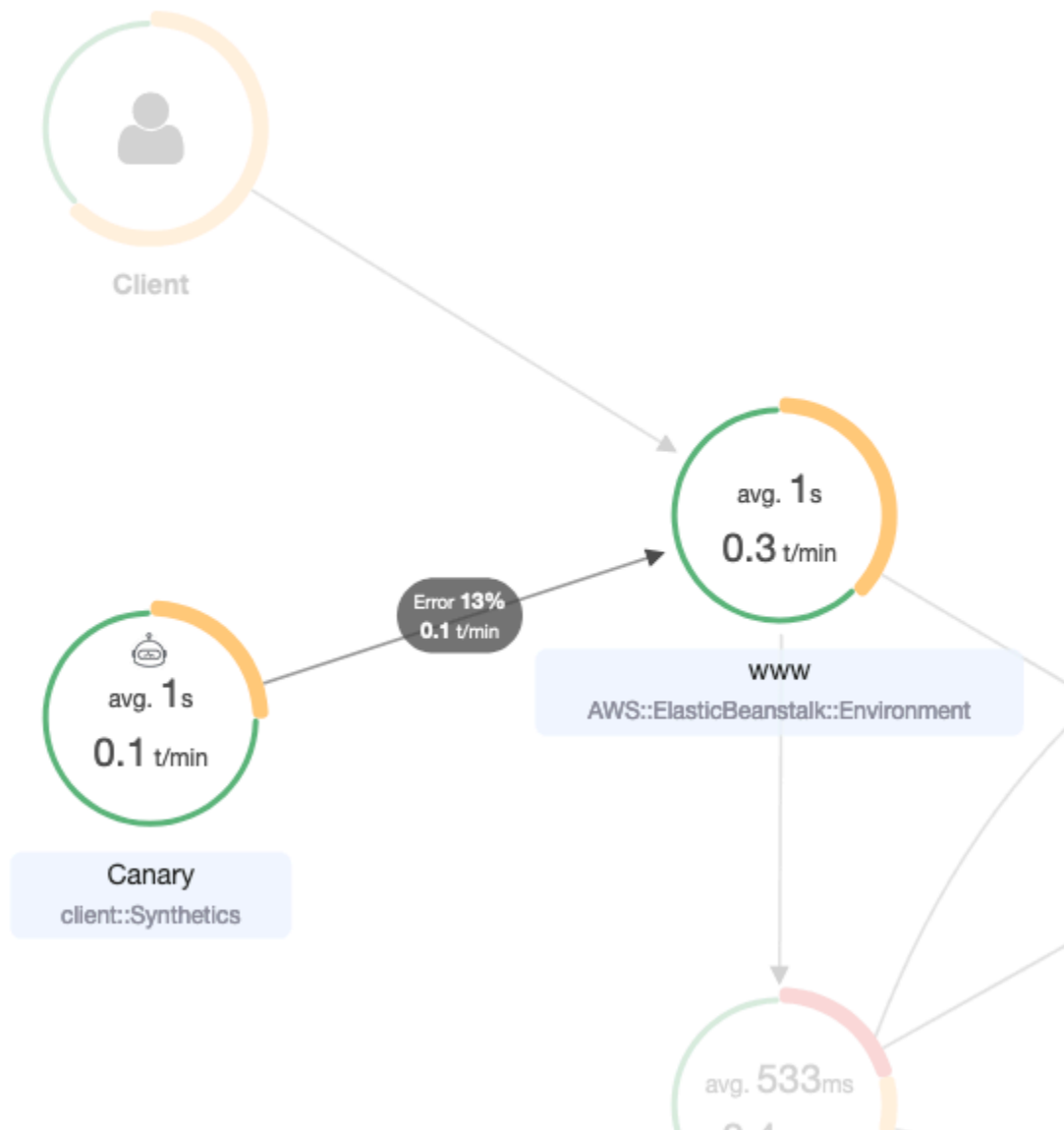


Usar grupos para se concentrar em testes do Synthetics

Você pode criar um grupo do X-Ray usando uma expressão de filtro para se concentrar em determinado conjunto de fluxos de trabalho, como testes do Synthetics para a aplicação “www” em execução no AWS Elastic Beanstalk. Use [palavras-chave complexas](#) `service()` e `edge()` para filtrar serviços e bordas.

Example Expressão do filtro de grupo

```
"edge(id(name: "www", type: "client::Synthetics"), id(name: "www", type:
"AWS::ElasticBeanstalk::Environment"))"
```



AWS Elastic Beanstalk and AWS X-Ray

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry Para obter mais

informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

AWS Elastic Beanstalk as plataformas incluem o daemon X-Ray. Você pode [executar o daemon](#) definindo uma opção no console do Elastic Beanstalk ou com um arquivo de configuração.

Na plataforma Java SE, você pode usar um arquivo Buildfile para criar seu aplicativo com Maven ou Gradle na instância. O X-Ray SDK for Java AWS SDK para Java está disponível no Maven, para que você possa implantar somente o código do seu aplicativo e criar na instância para evitar agrupar e carregar todas as suas dependências.

Você pode usar as propriedades do ambiente do Elastic Beanstalk para configurar o X-Ray SDK. O método que o Elastic Beanstalk usa para passar as propriedades do ambiente para a aplicação varia de acordo com a plataforma. Use as variáveis de ambiente do X-Ray SDK ou as propriedades do sistema de acordo com sua plataforma.

- [Plataforma Node.js](#): use [variáveis de ambiente](#).
- [Plataforma Java SE](#): use [variáveis de ambiente](#).
- [Plataforma Tomcat](#): use [propriedades do sistema](#).

Para obter mais informações, consulte [Configurando a AWS X-Ray depuração](#) no Guia do desenvolvedor. AWS Elastic Beanstalk

Elastic Load Balancing e AWS X-Ray

Os Application Load Balancers do Elastic Load Balancing adicionam um ID de rastreamento às solicitações HTTP de entrada em um cabeçalho chamado X-Amzn-Trace-Id.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Formato do ID de rastreamento do X-Ray

Um `trace_id` do X-Ray consiste em três números separados por hifens. Por exemplo, `.1-58406520-a006649127e371903a2de979` Isso inclui:

- O número da versão, que é 1.
- A hora da solicitação original, em horário epoch Unix, com 8 dígitos hexadecimais.

Por exemplo, 10h no dia 1º de dezembro de 2016 PST equivale a 1480615200 segundos em horário epoch ou a 58406520 em dígitos hexadecimais.

- Um identificador globalmente exclusivo de 96 bits para o rastreamento com 24 dígitos hexadecimais.

Os balanceadores de carga não enviam dados ao X-Ray e não são exibidos como um nó em seu mapa de serviço.

Para mais informações, consulte [Solicitar rastreamento para seu Application Load Balancer](#) no Guia do desenvolvedor do Elastic Load Balancing.

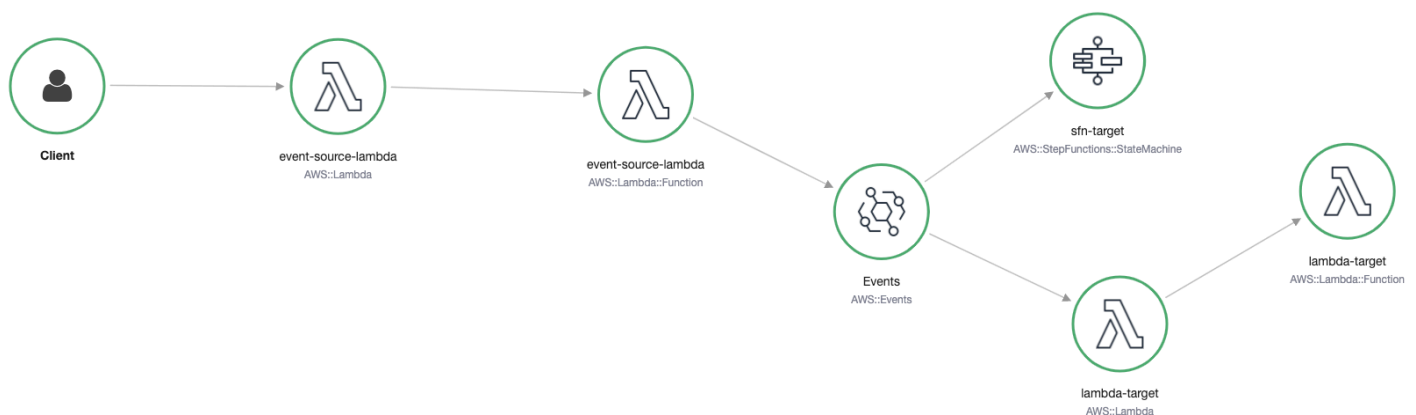
Amazon EventBridge e AWS X-Ray

AWS X-Ray se integra à Amazon EventBridge para rastrear eventos que são transmitidos EventBridge. [Se um serviço que é instrumentado com o X-Ray SDK envia eventos para EventBridge, o contexto de rastreamento é propagado para destinos de eventos downstream dentro do cabeçalho de rastreamento.](#) O X-Ray SDK pega automaticamente o cabeçalho de rastreamento e o aplica a qualquer instrumentação subsequente. Essa continuidade permite que os usuários rastreiem, analisem e depurem todos os serviços subsequentes e oferece uma visão mais completa do sistema.

Para obter mais informações, consulte [EventBridge X-Ray Integration](#) no Guia EventBridge do usuário.

Visualizar a origem e os destinos no mapa de serviço do X-Ray

O [mapa de rastreamento](#) X-Ray exibe um nó de EventBridge evento que conecta os serviços de origem e destino, como no exemplo a seguir:



Propagar o contexto de rastreamento para destinos de eventos

O X-Ray SDK permite que a fonte do EventBridge evento propague o contexto de rastreamento para destinos de eventos posteriores. Os exemplos específicos da linguagem a seguir demonstram a chamada EventBridge de uma função Lambda na qual o rastreamento [ativo](#) está habilitado:

Java

Adicione as dependências necessárias para o X-Ray:

- [AWS X-Ray SDK para Java](#)
- [AWS X-Ray SDK do gravador para Java](#)

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.services.eventbridge.AmazonEventBridge;
import com.amazonaws.services.eventbridge.AmazonEventBridgeClientBuilder;
import com.amazonaws.services.eventbridge.model.PutEventsRequest;
import com.amazonaws.services.eventbridge.model.PutEventsRequestEntry;
import com.amazonaws.services.eventbridge.model.PutEventsResult;
import com.amazonaws.services.eventbridge.model.PutEventsResultEntry;
import com.amazonaws.xray.handlers.TracingHandler;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.lang.StringBuilder;
import java.util.Map;
import java.util.List;
import java.util.Date;
import java.util.Collections;

/*
    Add the necessary dependencies for XRay:
    https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-xray
    https://mvnrepository.com/artifact/com.amazonaws/aws-xray-recorder-sdk-aws-sdk
*/
public class Handler implements RequestHandler<SQSEvent, String>{
```



```

private static final Logger logger = LoggerFactory.getLogger(Handler.class);

/*
    build EventBridge client
*/
private static final AmazonEventBridge eventsClient =
AmazonEventBridgeClientBuilder
    .standard()
    // instrument the EventBridge client with the XRay Tracing Handler.
    // the AWSXRay globalRecorder will retrieve the tracing-context
    // from the lambda function and inject it into the HTTP header.
    // be sure to enable 'active tracing' on the lambda function.
    .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
    .build();

@Override
public String handleRequest(SQSEvent event, Context context)
{
    PutEventsRequestEntry putEventsRequestEntry0 = new PutEventsRequestEntry();
    putEventsRequestEntry0.setTime(new Date());
    putEventsRequestEntry0.setSource("my-lambda-function");
    putEventsRequestEntry0.setDetailType("my-lambda-event");
    putEventsRequestEntry0.setDetail("{\"lambda-source\":\"sqs\"}");
    PutEventsRequest putEventsRequest = new PutEventsRequest();
    putEventsRequest.setEntries(Collections.singletonList(putEventsRequestEntry0));
    // send the event(s) to EventBridge
    PutEventsResult putEventsResult = eventsClient.putEvents(putEventsRequest);
    try {
        logger.info("Put Events Result: {}", putEventsResult);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "success";
}
}

```

Python

Adicione a seguinte dependência ao arquivo requirements.txt:

```
aws-xray-sdk==2.4.3
```

```
import boto3
```

```

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

# apply the XRay handler to all clients.
patch_all()

client = boto3.client('events')

def lambda_handler(event, context):
    response = client.put_events(
        Entries=[
            {
                'Source': 'foo',
                'DetailType': 'foo',
                'Detail': '{"foo": "foo"}'
            },
        ]
    )
    return response

```

Go

```

package main

import (
    "context"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-xray-sdk-go/xray"
    "github.com/aws/aws-sdk-go/service/eventbridge"
    "fmt"
)

var client = eventbridge.New(session.New())

func main() {
    //Wrap the eventbridge client in the AWS XRay tracer
    xray.AWS(client.Client)
    lambda.Start(handleRequest)
}

```

```

func handleRequest(ctx context.Context, event events.SQSEvent) (string, error) {
    _, err := callEventBridge(ctx)
    if err != nil {
        return "ERROR", err
    }
    return "success", nil
}

func callEventBridge(ctx context.Context) (string, error) {
    entries := make([]*eventbridge.PutEventsRequestEntry, 1)
    detail := "{ \"foo\": \"foo\"}"
    detailType := "foo"
    source := "foo"
    entries[0] = &eventbridge.PutEventsRequestEntry{
        Detail: &detail,
        DetailType: &detailType,
        Source: &source,
    }

    input := &eventbridge.PutEventsInput{
        Entries: entries,
    }

    // Example sending a request using the PutEventsRequest method.
    resp, err := client.PutEventsWithContext(ctx, input)

    success := "yes"
    if err == nil { // resp is now filled
        success = "no"
        fmt.Println(resp)
    }
    return success, err
}

```

Node.js

```

const AWSXRay = require('aws-xray-sdk')
//Wrap the aws-sdk client in the AWS XRay tracer
const AWS = AWSXRay.captureAWS(require('aws-sdk'))
const eventBridge = new AWS.EventBridge()

exports.handler = async (event) => {

```

```

let myDetail = { "name": "Alice" }

const myEvent = {
  Entries: [{
    Detail: JSON.stringify({ myDetail }),
    DetailType: 'myDetailType',
    Source: 'myApplication',
    Time: new Date
  }]
}

// Send to EventBridge
const result = await eventBridge.putEvents(myEvent).promise()

// Log the result
console.log('Result: ', JSON.stringify(result, null, 2))
}

```

C#

Adicione os seguintes pacotes do X-Ray às dependências do C#:

```

<PackageReference Include="AWSXRayRecorder.Core" Version="2.6.2" />
<PackageReference Include="AWSXRayRecorder.Handlers.AwsSdk" Version="2.7.2" />

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Amazon;
using Amazon.Util;
using Amazon.Lambda;
using Amazon.Lambda.Model;
using Amazon.Lambda.Core;
using Amazon.EventBridge;
using Amazon.EventBridge.Model;
using Amazon.Lambda.SQSEvents;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.AwsSdk;
using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;

```

```
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.Json.JsonSerializer))]

namespace blankCsharp
{
    public class Function
    {
        private static AmazonEventBridgeClient eventClient;

        static Function() {
            initialize();
        }

        static async void initialize() {
            //Wrap the AWS SDK clients in the AWS XRay tracer
            AWSSDKHandler.RegisterXRayForAllServices();
            eventClient = new AmazonEventBridgeClient();
        }

        public async Task<PutEventsResponse> FunctionHandler(SQSEvent invocationEvent,
            ILambdaContext context)
        {
            PutEventsResponse response;
            try
            {
                response = await callEventBridge();
            }
            catch (AmazonLambdaException ex)
            {
                throw ex;
            }

            return response;
        }

        public static async Task<PutEventsResponse> callEventBridge()
        {
            var request = new PutEventsRequest();
            var entry = new PutEventsRequestEntry();
            entry.DetailType = "foo";
            entry.Source = "foo";
            entry.Detail = "{\"instance_id\":\"A\"}";
            List<PutEventsRequestEntry> entries = new List<PutEventsRequestEntry>();
```

```
entries.Add(entry);
request.Entries = entries;
var response = await eventClient.PutEventsAsync(request);
return response;
}
}
```

AWS Lambda and AWS X-Ray

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Você pode usar AWS X-Ray para rastrear suas AWS Lambda funções. O Lambda executa o [daemon do X-Ray](#) e registra um segmento com detalhes sobre como invocar e executar a função. Para ampliar a instrumentação, você pode empacotar o X-Ray SDK com sua função para registrar chamadas de saída e adicionar anotações e metadados.

Se a função do Lambda for chamada por outro serviço instrumentado, o Lambda rastreará as solicitações que já foram amostradas sem nenhuma configuração adicional. O serviço precedente pode ser uma aplicação web instrumentada ou outra função do Lambda. Seu serviço pode invocar a função diretamente com um cliente AWS SDK instrumentado ou chamando uma API do API Gateway com um cliente HTTP instrumentado.

AWS X-Ray suporta o rastreamento de aplicativos orientados por eventos usando o Amazon AWS Lambda SQS. Use o CloudWatch console para ver uma visão conectada de cada solicitação conforme ela é enfileirada com o Amazon SQS e processada por uma função Lambda downstream. Os rastreamentos dos produtores de mensagens upstream são automaticamente vinculados aos rastreamentos dos nós consumidores do Lambda downstream, criando uma end-to-end visão do aplicativo. Para obter mais informações, consulte [Rastrear aplicativos orientadas a eventos](#).

 Note

Se você tiver rastreamentos habilitados para uma função Lambda downstream, também deverá habilitar rastreamentos para a função Lambda raiz que chama a função downstream para que a função downstream gere rastreamentos.

Se a função do Lambda for executada de acordo com uma programação ou invocada por um serviço que não está instrumentado, você poderá configurar o Lambda para coletar amostras e registrar invocações por meio do rastreamento ativo.

Para configurar a integração do X-Ray em uma AWS Lambda função

1. Abra o [console do AWS Lambda](#).
2. Selecione Funções no painel de navegação à esquerda.
3. Escolha a função.
4. Na guia Configuração, role para baixo até o cartão Ferramentas adicionais de monitoramento. Você também pode encontrar esse cartão selecionando Ferramentas de monitoramento e operações no painel de navegação esquerdo.
5. Selecione Editar.
6. Em AWS X-Ray, habilite o Rastreamento ativo.

Em runtimes com um X-Ray SDK correspondente, o Lambda também executa o daemon do X-Ray.

X-Ray SDKs em Lambda

- X-Ray SDK para Go: Go 1.7 e runtimes mais recentes
- X-Ray SDK para Java: runtime Java 8
- X-Ray SDK para Node.js: Node.js 4.3 e runtimes mais recentes
- X-Ray SDK para Python: Python 2.7, Python 3.6 e runtimes mais novos
- X-Ray SDK para .NET: .NET Core 2.0 e runtimes mais recentes

Para usar o X-Ray SDK no Lambda, empacote-o com o código da função toda vez que você criar uma versão. É possível instrumentar as funções do Lambda com os mesmos métodos usados para instrumentar aplicações em execução em outros serviços. A principal diferença é que você não usa o SDK para instrumentar solicitações de entrada, tomar decisões de amostragem e criar segmentos.

A outra diferença entre a instrumentação de funções do Lambda e de aplicações web é que o segmento criado pelo Lambda e enviado para o X-Ray não pode ser modificado pelo código da função. Você pode criar subsegmentos e gravar anotações e metadados neles, mas não é possível adicionar anotações e metadados ao segmento pai.

Para obter mais informações, consulte [Usar o AWS X-Ray](#) no Guia do desenvolvedor do AWS Lambda .

AWS Step Functions e AWS X-Ray

O AWS X-Ray se integra ao AWS Step Functions para rastrear e analisar solicitações para o Step Functions. Você pode visualizar os componentes da aplicação, identificar gargalos de performance e solucionar problemas de solicitações que resultaram em um erro. Para obter mais informações, consulte [AWS X-Ray e Step Functions](#) no Guia do desenvolvedor do AWS Step Functions.

Como habilitar o rastreamento do X-Ray ao criar uma máquina de estado

1. Abra o console do Step Functions em <https://console.aws.amazon.com/states/>.
2. Selecione Criar máquina de estado.
3. Na página Definir máquina de estado, selecione Criar com trechos de código. Se você optar por executar um projeto de exemplo, não poderá habilitar o rastreamento do X-Ray durante a criação. Em vez disso, habilite o rastreamento do X-Ray depois de criar a máquina de estado.
4. Escolha Próximo.
5. Na página Especificar detalhes, configure a máquina de estado.
6. Escolha Habilitar rastreamento do X-Ray.

Como habilitar o rastreamento do X-Ray em uma máquina de estado existente

1. No console do Step Functions, selecione a máquina de estado para a qual você deseja habilitar o rastreamento.
2. Escolha Editar.
3. Escolha Ativar rastreamento do X-Ray.
4. (Opcional) Gere automaticamente um novo perfil para sua máquina de estado para incluir permissões do X-Ray escolhendo Criar novo perfil na janela Permissões.

Permissions**Execution role**

The IAM role that defines which resources your state machine has permission to access during execution. To create a custom role, go to the [IAM console](#).

☒ **Create new role**

[Let Step Functions create a new role for you based on your state machine's definition and configuration details.](#)

☐ Choose an existing role

☐ Enter a role ARN

5. Escolha Salvar.

Note

Quando você cria uma nova máquina de estado, ela é rastreada automaticamente se a solicitação for amostrada e o rastreamento estiver habilitado em um serviço precedente, como o Amazon API Gateway ou o AWS Lambda. Para qualquer máquina de estado existente não configurada por meio do console (por exemplo, por meio de um modelo do CloudFormation), verifique se você tem uma política do IAM que conceda permissões suficientes para habilitar rastreamentos do X-Ray.

Instrumentando seu aplicativo para AWS X-Ray

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

A instrumentação de um aplicativo requer o envio de dados de rastreamento para solicitações de entrada e saída e outros eventos dentro do aplicativo, bem como metadados sobre cada solicitação. Há várias opções de instrumentação diferentes que você pode escolher ou combinar, com base em seus aplicativos específicas:

- Instrumentação automática: instrumente seu aplicativo sem alterações no código, normalmente por meio de alterações de configuração, adição de um agente de instrumentação automática ou outros mecanismos.
- Instrumentação de biblioteca — faça alterações mínimas no código do aplicativo para adicionar instrumentação pré-criada voltada para bibliotecas ou estruturas específicas, como o AWS SDK, clientes Apache HTTP ou clientes SQL.
- Instrumentação manual: adicione o código de instrumentação ao aplicativo em cada local para o qual você deseja enviar informações de rastreamento.

Existem vários SDKs agentes e ferramentas que podem ser usados para instrumentar seu aplicativo para rastreamento de raios-X.

Tópicos

- [Instrumentando seu aplicativo com a AWS Distro para OpenTelemetry](#)
- [Instrumentando seu aplicativo com AWS X-Ray SDKs](#)
- [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#)

Instrumentando seu aplicativo com a AWS Distro para OpenTelemetry

A AWS Distro for OpenTelemetry (ADOT) é uma AWS distribuição baseada no projeto Cloud Native Computing Foundation (CNCF). OpenTelemetry OpenTelemetry fornece um único conjunto de código aberto APIs, bibliotecas e agentes para coletar rastreamentos e métricas distribuídos. Este kit de ferramentas é uma distribuição de OpenTelemetry componentes iniciais SDKs, incluindo agentes de instrumentação automática e coletores que são testados, otimizados, protegidos e suportados pela AWS.

Com o ADOT, os engenheiros podem instrumentar seus aplicativos uma vez e enviar métricas e rastreamentos correlacionados para várias soluções de AWS monitoramento CloudWatch AWS X-Ray, incluindo Amazon e Amazon OpenSearch Service.

O uso do X-Ray com o ADOT requer dois componentes: um OpenTelemetry SDK habilitado para uso com o X-Ray e o AWS Distro for OpenTelemetry Collector habilitado para uso com o X-Ray. Para obter mais informações sobre como usar a AWS Distro for OpenTelemetry with AWS X-Ray e other Serviços da AWS, consulte a [AWS Distro for OpenTelemetry Documentation](#).

Para obter mais informações sobre suporte e uso de idiomas, consulte [AWS Observabilidade ativada GitHub](#).

Note

Agora você pode usar o CloudWatch agente para coletar métricas, registros e rastreamentos de EC2 instâncias da Amazon e servidores locais. CloudWatch O agente versão 1.300025.0 e posterior pode coletar traços de nosso cliente [OpenTelemetryX-Ray](#) e enviá-los para o [SDKsX-Ray](#). Usar o CloudWatch agente em vez do AWS Distro for OpenTelemetry (ADOT) Collector ou do daemon X-Ray para coletar traços pode ajudar a reduzir o número de agentes que você gerencia. Consulte o tópico do [CloudWatch agente](#) no Guia do CloudWatch usuário para obter mais informações.

A tabela inclui o seguinte:

- [AWS Distro para Go OpenTelemetry](#)
- [AWS Distro para Java OpenTelemetry](#)
- [AWS Distro para OpenTelemetry JavaScript](#)

- [AWS Distro para Python OpenTelemetry](#)
- [AWS Distro para .NET OpenTelemetry](#)

No momento, o ADOT comporta instrumentação automática para [Java](#) e [Python](#). [Além disso, o ADOT permite a instrumentação automática das funções do AWS Lambda e suas solicitações downstream usando tempos de execução de Java, Node.js e Python, por meio do ADOT Managed Lambda Layers.](#)

O ADOT SDKs para Java e Go oferece suporte às regras de amostragem centralizada do X-Ray. Se você precisar de suporte para as regras de amostragem do X-Ray em outros idiomas, considere usar um AWS X-Ray SDK.

Note

Você pode enviar agora o rastreamento W3C para o IDs X-Ray. Por padrão, os rastreamentos criados com OpenTelemetry têm um formato de ID de rastreamento baseado na especificação do [W3C Trace Context](#). Isso é diferente do formato de rastreamento IDs criado usando um X-Ray SDK ou por AWS serviços integrados ao X-Ray. [Para garantir que o rastreamento IDs no formato W3C seja aceito pelo X-Ray, você deve usar a versão 0.86.0 ou posterior do AWS X-Ray Exporter, incluída na versão 0.34.0 e posterior do ADOT Collector.](#) As versões anteriores do exportador validam os timestamps de ID de rastreamento, o que pode fazer com que o rastreamento W3C seja rejeitado. IDs

Instrumentando seu aplicativo com AWS X-Ray SDKs

AWS X-Ray inclui um conjunto de linguagens específicas SDKs para instrumentar seu aplicativo para enviar rastreamentos ao X-Ray. Cada X-Ray SDK fornece o seguinte:

- Interceptadores a serem adicionados ao código para rastrear solicitações HTTP recebidas
- Manipuladores de clientes para AWS instrumentar clientes SDK que seu aplicativo usa para chamar outros Serviços da AWS
- Um cliente HTTP para instrumentar chamadas para outros serviços da web HTTP internos e externos

O X-Ray SDKs também oferece suporte a chamadas de instrumentação para bancos de dados SQL, instrumentação automática de clientes AWS SDK e outros recursos. Em vez de enviar dados

de rastreamento diretamente ao X-Ray, os SDK enviam documentos de segmentos JSON a um processo do daemon que escuta o tráfego UDP. O [daemon do X-Ray](#) armazena os segmentos em buffer em uma fila e os carrega em lote no X-Ray.

São fornecidos os seguintes idiomas específicos SDKs :

- [AWS X-Ray SDK para Go](#)
- [AWS X-Ray SDK para Java](#)
- [AWS X-Ray SDK para Node.js](#)
- [AWS X-Ray SDK para Python](#)
- [AWS X-Ray SDK para .NET](#)
- [AWS X-Ray SDK para Ruby](#)

No momento, o X-Ray comporta instrumentação automática para [Java](#).

Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs

Os SDKs incluídos no X-Ray são parte de uma solução de instrumentação altamente integrada oferecida pela AWS. O AWS Distro OpenTelemetry for faz parte de uma solução mais ampla do setor, na qual o X-Ray é apenas uma das muitas soluções de rastreamento. Você pode implementar o end-to-end rastreamento no X-Ray usando qualquer uma das abordagens, mas é importante entender as diferenças para determinar a abordagem mais útil para você.

Recomendamos instrumentar seu aplicativo com a AWS Distro OpenTelemetry se você precisar do seguinte:

- Capacidade de enviar rastreamentos para vários backends de rastreamento diferentes sem precisar reinstrumentar o código
- Support para um grande número de instrumentações de biblioteca para cada idioma, mantidas pela comunidade OpenTelemetry
- Camadas do Lambda totalmente gerenciadas que empacotam tudo o que você precisa para coletar dados de telemetria, sem exigir alterações de código ao usar Java, Python ou Node.js.

 Note

AWS O Distro for OpenTelemetry oferece uma experiência inicial mais simples para instrumentar suas funções do Lambda. No entanto, devido às OpenTelemetry ofertas de flexibilidade, sua função Lambda exigirá memória adicional e as invocações podem sofrer aumentos de latência de inicialização a frio, o que pode levar a cobranças adicionais. Se você está otimizando para baixa latência e não precisa de recursos avançados, como destinos OpenTelemetry de back-end configuráveis dinamicamente, convém usar o AWS X-Ray SDK para instrumentar seu aplicativo.

Recomendamos escolher um X-Ray SDK para instrumentar o aplicativo se você precisar do seguinte:

- Uma solução totalmente integrada de um único fornecedor.
- Integração com as regras de amostragem centralizada do X-Ray, incluindo a capacidade de configurar regras de amostragem por meio do console do X-Ray e usá-las automaticamente em vários hosts, ao utilizar Node.js, Python, Ruby ou .NET

Transaction Search

O Transaction Search é uma experiência de analytics interativo que você pode usar para obter visibilidade completa das extensões de transação do aplicativo. As extensões são as unidades fundamentais de operação em um rastreamento distribuído e representam ações ou tarefas específicas em um aplicativo ou sistema. Cada extensão registra detalhes sobre um segmento específico da transação. Esses detalhes incluem horários de início e término, duração e metadados associados, que podem incluir atributos comerciais, como cliente IDs e pedido IDs. As extensões são organizadas em uma hierarquia pai-filho. Essa hierarquia forma um rastreamento completo, mapeando o fluxo de uma transação em diferentes componentes ou serviços.

Para obter mais informações, consulte [Transaction Search](#).

OpenTelemetry Ponto final do protocolo (OTLP)

OpenTelemetry é uma estrutura de observabilidade de código aberto que fornece às equipes de TI protocolos e ferramentas padronizados para coletar e rotear dados de telemetria. Ele fornece um formato unificado para instrumentar, gerar, reunir e exportar dados de telemetria de aplicativos, como métricas, logs e rastreamentos, para plataformas de monitoramento para fins de análise e insights. Ao usar OpenTelemetry, as equipes podem evitar a dependência de fornecedores, garantindo flexibilidade em suas soluções de observabilidade.

Você pode usar OpenTelemetry para enviar rastreamentos diretamente para um endpoint de OpenTelemetry protocolo (OTLP) e obter experiências de monitoramento de desempenho de out-of-the aplicativos em [CloudWatch Application](#) Signals.

Para obter mais informações, consulte [OpenTelemetry](#).

Trabalho com Go

Há duas maneiras de instrumentar uma aplicação Go para enviar rastreamento ao X-Ray:

- [AWS Distro para OpenTelemetry Go](#): uma distribuição da AWS que fornece um conjunto de bibliotecas de código aberto para enviar métricas e rastreamentos correlacionados para várias soluções de monitoramento da AWS, como o Amazon CloudWatch, o AWS X-Ray e o Amazon OpenSearch Service, por meio do [coletor AWS Distro para OpenTelemetry](#).
- [AWS X-Ray SDK para Go](#): um conjunto de bibliotecas para gerar e enviar rastreamentos ao X-Ray por meio do [daemon do X-Ray](#).

Para obter mais informações, consulte [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#).

AWS Distro para Go OpenTelemetry

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Com o AWS Distro for OpenTelemetry Go, você pode instrumentar seus aplicativos uma vez e enviar métricas e rastreamentos correlacionados para várias soluções de AWS monitoramento CloudWatch AWS X-Ray, incluindo Amazon e Amazon OpenSearch Service. O uso do X-Ray com o AWS Distro for OpenTelemetry requer dois componentes: um OpenTelemetry SDK habilitado para uso com o X-Ray e o AWS Distro for OpenTelemetry Collector habilitado para uso com o X-Ray.

Para começar, consulte a [documentação do AWS Distro for OpenTelemetry Go](#).

Para obter mais informações sobre como usar a AWS Distro for OpenTelemetry with AWS X-Ray e other Serviços da AWS, consulte [AWS Distro for OpenTelemetry ou Distro for AWS Documentation](#). OpenTelemetry

Para obter mais informações sobre suporte e uso de idiomas, consulte [AWS Observabilidade ativada GitHub](#).

AWS X-Ray SDK para Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

O X-Ray SDK para Go é um conjunto de bibliotecas para aplicativos Go que fornece classes e métodos para gerar e enviar dados de rastreamento ao daemon do X-Ray SDK. Os dados de rastreamento incluem informações sobre solicitações HTTP recebidas pelo aplicativo e chamadas que o aplicativo faz para serviços downstream usando o AWS SDK, clientes HTTP ou um conector de banco de dados SQL. Você também pode criar segmentos manualmente e adicionar informações de depuração em anotações e metadados.

Faça o download do SDK de seu [GitHubrepositório](#) com: `go get`

```
$ go get -u github.com/aws/aws-xray-sdk-go/...
```

Para aplicativos web, comece [usando a xray.Handlerfunção](#) para rastrear solicitações de entrada. O manipulador de mensagens cria um [segmento](#) para cada solicitação rastreada e conclui o segmento quando a resposta é enviada. Embora o segmento esteja aberto, você pode usar os métodos do cliente do SDK para adicionar informações ao segmento e criar subsegmentos para rastrear as chamadas subsequentes. O SDK também registra automaticamente exceções que seu aplicativo lança enquanto o segmento está aberto.

Para funções do Lambda chamadas por um aplicativo ou um serviço instrumentado, o Lambda lê o [cabeçalho de rastreamento](#) e rastreia automaticamente as solicitações amostradas. Para outras funções, você pode [configurar o Lambda](#) para amostrar e rastrear solicitações recebidas. Em ambos os casos, o Lambda cria o segmento e o fornece ao X-Ray SDK.

Note

No Lambda, o X-Ray SDK é opcional. Se você não o usar em sua função, mesmo assim o mapa de serviço incluirá um nó para o serviço Lambda e um para cada função do Lambda. Ao adicionar o SDK, você pode instrumentar o código da função para adicionar subsegmentos ao segmento de função registrado pelo Lambda. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Em seguida, [encapsule seu cliente com uma chamada para a função AWS](#). Essa etapa garante que os instrumentos do X-Ray chamem qualquer método de cliente. Você também pode [instrumentar chamadas para bancos de dados SQL](#).

Depois que você começar a usar o SDK, personalize o comportamento dele [configurando o gravador e o middleware](#). Você pode adicionar plug-ins para registrar dados sobre os recursos de computação que executam seu aplicativo, personalizar o comportamento de amostragem, estipulando regras de amostragem, e definir o nível de log para ver mais ou menos informações do SDK nos logs do aplicativo.

Registre informações adicionais sobre as solicitações e o trabalho que o aplicativo faz em [anotações e metadados](#). Anotações são simples pares de chave-valor que são indexados para serem usados com [expressões de filtro](#) para que você possa pesquisar rastreamentos que contêm dados específicos. As entradas de metadados são menos restritivas e podem registrar matrizes e objetos inteiros: tudo o que pode ser serializado em JSON.

Anotações e metadados

Anotações e metadados são textos arbitrários que você adiciona aos segmentos com o X-Ray SDK. As anotações são indexadas para serem usadas com expressões de filtro. Os metadados não são indexados, mas podem ser visualizados no segmento bruto com o console ou a API do X-Ray. Qualquer pessoa à qual você conceder acesso de leitura ao X-Ray poderá visualizar esses dados.

Quando há uma grande quantidade de clientes instrumentados no código, um único segmento de solicitação pode conter um grande número de subsegmentos, um para cada chamada feita com um cliente instrumentado. Você pode organizar e agrupar subsegmentos integrando chamadas de clientes em [subsegmentos personalizados](#). Você pode criar um subsegmento personalizado para uma função inteira ou qualquer seção de código e registrar metadados e anotações no subsegmento em vez de gravar tudo no segmento principal.

Requisitos

O X-Ray SDK para Go requer o Go 1.9 ou posterior.

O SDK depende das seguintes bibliotecas para compilação e tempo de execução:

- AWS SDK for Go versão 1.10.0 ou mais recente

Essas dependências são declaradas no arquivo `README.md` do SDK.

Documentação de referência

Depois de baixar o SDK, crie e hospede a documentação localmente para visualizá-la em um navegador da web.

Para visualizar a documentação de referência

1. Navegando para o diretório `$GOPATH/src/github.com/aws/aws-xray-sdk-go` (Linux ou Mac) ou para a pasta `%GOPATH%\src\github.com\aws\aws-xray-sdk-go` (Windows)
2. Execute o comando `godoc`.

```
$ godoc -http=:6060
```

3. Abrindo um navegador em `http://localhost:6060/pkg/github.com/aws/aws-xray-sdk-go/`.

Configurar o X-Ray SDK para Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-

Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode especificar a configuração do X-Ray SDK para Go por meio de variáveis de ambiente chamando Configure com um objeto Config ou assumindo valores padrão. As variáveis de ambiente têm prioridade sobre os valores de Config, que têm precedência sobre qualquer valor padrão.

Seções

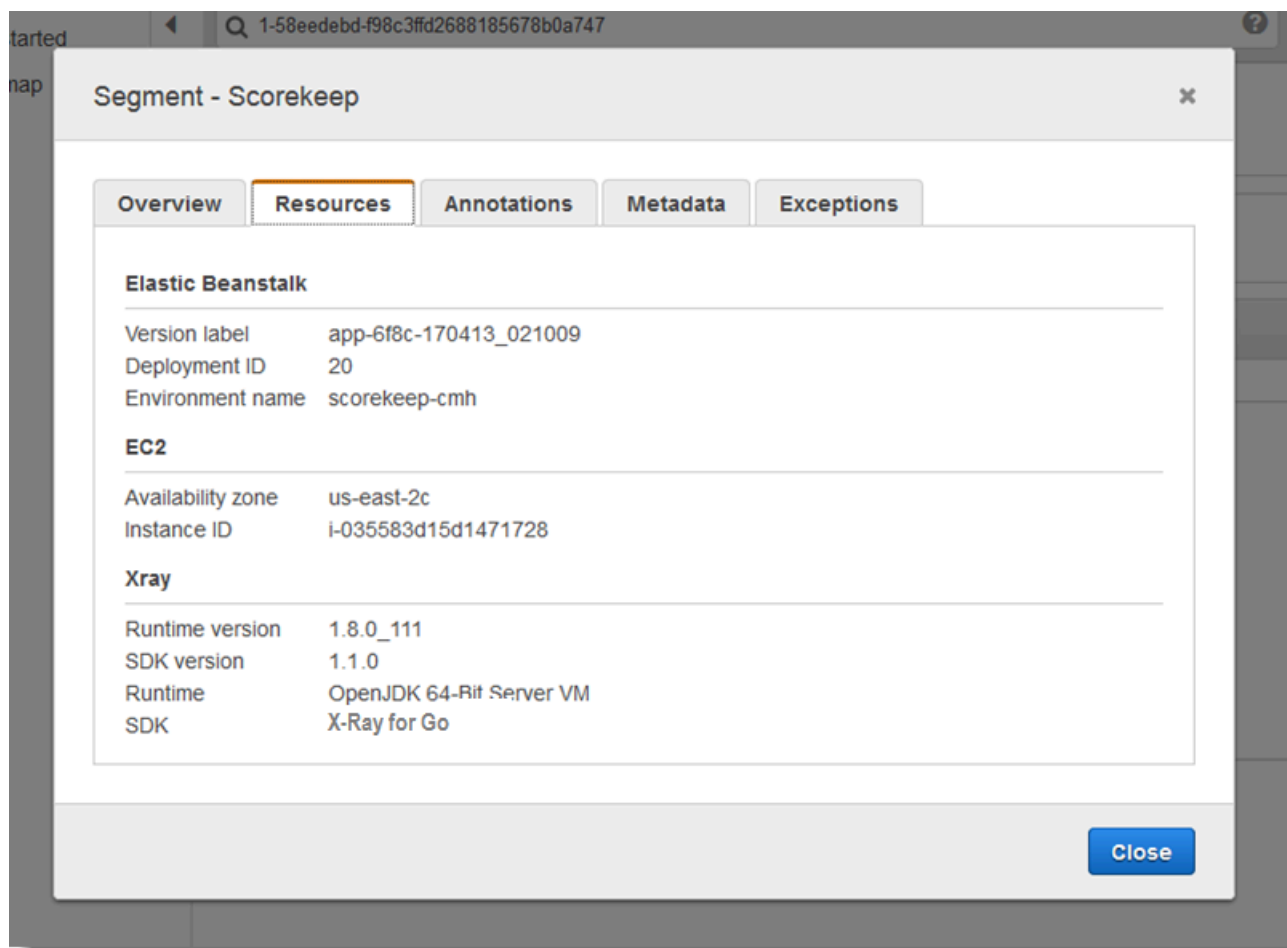
- [Plug-ins de serviço](#)
- [Regras de amostragem](#)
- [Registro em log](#)
- [Variáveis de ambiente](#)
- [Usar Configure](#)

Plug-ins de serviço

Use plugins para registrar informações sobre o serviço que hospeda o aplicativo.

Plugins

- Amazon EC2 — EC2Plugin adiciona o ID da instância, a zona de disponibilidade e o grupo de CloudWatch registros.
- Elastic Beanstalk: o ElasticBeanstalkPlugin adiciona o nome do ambiente, o rótulo da versão e o ID de implantação.
- Amazon ECS: o ECSPlugin adiciona o ID do contêiner.



Para usar um plug-in, importe um dos seguintes pacotes.

```
"github.com/aws/aws-xray-sdk-go/awsplugins/ec2"
"github.com/aws/aws-xray-sdk-go/awsplugins/ecs"
"github.com/aws/aws-xray-sdk-go/awsplugins/beanstalk"
```

Cada plug-in tem uma chamada de função `Init()` explícita que carrega o plug-in.

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",
})
}
```

O SDK também usa as configurações do plug-in para definir o campo `origin` no segmento. Isso indica o tipo de AWS recurso que executa seu aplicativo. Quando você usa vários plug-ins, o SDK usa a seguinte ordem de resolução para determinar a origem: ElasticBeanstalk > EKS > ECS > EC2

Regras de amostragem

O SDK usa as regras de amostragem que você define no console do X-Ray para determinar quais solicitações serão registradas. A regra padrão rastreia a primeira solicitação a cada segundo e 5% de todas as solicitações adicionais em todos os serviços que enviam rastreamentos ao X-Ray. [Crie regras adicionais no console do X-Ray](#) para personalizar a quantidade de dados registrados para cada uma dos aplicativos.

O SDK aplica regras personalizadas na ordem em que elas estão definidas. Se uma solicitação corresponder a várias regras personalizadas, o SDK aplicará somente a primeira regra.

Note

Se o SDK não conseguir acessar o X-Ray para obter regras de amostragem, ele reverterá para uma regra local padrão da primeira solicitação a cada segundo e 5% de todas as solicitações adicionais por host. Isso pode ocorrer se o host não tiver permissão para chamar a amostragem APIs ou não conseguir se conectar ao daemon X-Ray, que atua como um proxy TCP para chamadas de API feitas pelo SDK.

Você também pode configurar o SDK para carregar regras de amostragem de um documento JSON. O SDK pode usar regras locais como backup para casos em que a amostragem do X-Ray não está disponível ou usar exclusivamente regras locais.

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
}
}
```

Este exemplo define uma regra personalizada e uma regra padrão. A regra personalizada aplica uma taxa de amostragem de 5% sem um número mínimo de solicitações para rastrear os caminhos em `/api/move/`. A regra padrão rastreia a primeira solicitação a cada segundo e 10% das solicitações adicionais.

A desvantagem de definir regras localmente é que o destino fixo é aplicado por instância do gravador de forma independente, em vez de ser gerenciado pelo serviço X-Ray. À medida que você implanta mais hosts, a taxa fixa é multiplicada, dificultando o controle da quantidade de dados registrados.

Ativado AWS Lambda, você não pode modificar a taxa de amostragem. Se sua função for chamada por um serviço instrumentado, as chamadas que geraram solicitações que foram amostradas por esse serviço serão registradas pelo Lambda. Se o rastreamento ativo estiver habilitado e nenhum cabeçalho de rastreamento estiver presente, o Lambda tomará a decisão de amostragem.

Para fornecer regras de backup, aponte para o arquivo JSON de amostragem local usando o `NewCentralizedStrategyWithFilePath`.

Example main.go: regra de amostragem local

```
s, _ := sampling.NewCentralizedStrategyWithFilePath("sampling.json") // path to local
sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Para usar somente regras locais, aponte para o arquivo JSON de amostragem local usando o `NewLocalizedStrategyFromFilePath`.

Example main.go: desabilitar a amostragem

```
s, _ := sampling.NewLocalizedStrategyFromFilePath("sampling.json") // path to local
sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Registro em log

Note

Os campos `xray.Config{}` `LogLevel` e `LogFormat` estão defasados a partir da versão 1.0.0-rc.10.

O X-Ray usa a interface a seguir para registro em log. O registrador em log padrão grava em `stdout` em `LogLevelInfo` e acima.

```
type Logger interface {
    Log(level LogLevel, msg fmt.Stringer)
}

const (
    LogLevelDebug LogLevel = iota + 1
    LogLevelInfo
    LogLevelWarn
    LogLevelError
)
```

Example gravar em `io.Writer`

```
xray.SetLogger(xraylog.NewDefaultLogger(os.Stderr, xraylog.LogLevelError))
```

Variáveis de ambiente

É possível usar variáveis de ambiente para configurar o X-Ray SDK para Go. O SDK é compatível com as variáveis a seguir.

- `AWS_XRAY_CONTEXT_MISSING`: defina como `RUNTIME_ERROR` para lançar exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.

Valores válidos

- `RUNTIME_ERROR`: lance uma exceção de tempo de execução.
- `LOG_ERROR`: registre um erro e continue (padrão).
- `IGNORE_ERROR`: ignore o erro e continue.

Erros relativos a segmentos ou subsegmentos ausentes poderão ocorrer quando você tentar usar um cliente instrumentado no código de inicialização que é executado quando nenhuma solicitação estiver aberta ou em um código que gere um novo thread.

- `AWS_XRAY_TRACING_NAME`: defina o nome do serviço que o SDK usa para segmentos.
- `AWS_XRAY_DAEMON_ADDRESS`: defina o host e a porta do receptor do daemon do X-Ray. Por padrão, o SDK envia dados de rastreamento para `127.0.0.1:2000`. Use essa variável se você tiver configurado o daemon para [escutar em uma porta diferente](#) ou se ele estiver sendo executado em um host diferente.
- `AWS_XRAY_CONTEXT_MISSING`: defina o valor para determinar como o SDK lida com erros de contexto ausente. Erros relativos a segmentos faltantes ou subsegmentos poderão ocorrer quando você tentar usar um cliente instrumentado no código de inicialização quando nenhuma solicitação estiver aberta, ou em um código que gere um novo thread.
 - `RUNTIME_ERROR`: por padrão, o SDK é definido para a lançar uma exceção de tempo de execução.
 - `LOG_ERROR`: defina para registrar um erro em log e continuar.

As variáveis de ambiente substituem os valores equivalentes definidos no código.

Usar Configure

Você também pode configurar o X-Ray SDK para Go usando o método `Configure`. O `Configure` usa um argumento, um objeto `Config`, com os campos opcionais a seguir.

DaemonAddr

Essa string especifica o host e a porta do receptor do daemon do X-Ray. Se não especificado, o X-Ray usará o valor da variável de ambiente `AWS_XRAY_DAEMON_ADDRESS`. Se esse valor não estiver definido, `"127.0.0.1:2000"` será usado.

ServiceVersion

Essa sequência de caracteres especifica a versão do serviço. Se não especificado, o X-Ray usará a string vazia ("").

SamplingStrategy

Esse objeto `SamplingStrategy` especifica quais de suas chamadas do aplicativo são rastreadas. Se não especificado, o X-Ray usará uma `LocalizedSamplingStrategy`, que usa a estratégia conforme definido em `xray/resources/DefaultSamplingRules.json`.

StreamingStrategy

Esse `StreamingStrategy` objeto especifica se um segmento deve ser transmitido quando `RequiresStreaming` retorna verdadeiro. Se não especificado, o X-Ray usará uma `DefaultStreamingStrategy`, que transmitirá um segmento amostrado se o número de subsegmentos for superior a 20.

ExceptionFormattingStrategy

Esse objeto `ExceptionFormattingStrategy` especifica como você deseja lidar com várias exceções. Se não especificado, o X-Ray usará uma `DefaultExceptionFormattingStrategy` com um `XrayError` do tipo `error`, a mensagem de erro e o rastreamento de pilha.

Instrumentar solicitações HTTP de entrada com o X-Ray SDK para Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode usar o X-Ray SDK para rastrear solicitações HTTP recebidas que seu aplicativo atende em uma EC2 instância na Amazon EC2 ou no Amazon AWS Elastic Beanstalk ECS.

Use `xray.Handler` para instrumentar solicitações HTTP de entrada. O X-Ray SDK para Go implementa a interface `http.Handler` da biblioteca padrão do Go na classe `xray.Handler` para interceptar solicitações da web. A classe `xray.Handler` encapsula o `http.Handler` fornecido com o `xray.Capture` usando o contexto da solicitação, analisando cabeçalhos de entrada, adicionando cabeçalhos de resposta, se necessário, e define campos de rastreamento específicos ao HTTP.

Quando você usa essa classe para lidar com as solicitações e respostas HTTP, o X-Ray SDK para Go cria um segmento para cada solicitação amostrada. Este segmento inclui tempo, método e disposição da solicitação HTTP. Instrumentação adicional cria subsegmentos sobre esse segmento.

Note

Para AWS Lambda funções, o Lambda cria um segmento para cada solicitação de amostra. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

O exemplo a seguir intercepta as solicitações na porta 8000 e retorna "Hello!" como uma resposta. Ele cria o segmento `myApp` e instrumenta as chamadas por meio de qualquer aplicativo.

Example `main.go`

```
func main() {
    http.Handle("/", xray.Handler(xray.NewFixedSegmentNamer("MyApp"),
    http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Hello!"))
    })))

    http.ListenAndServe(":8000", nil)
}
```

Cada segmento tem um nome que identifica o aplicativo no mapa de serviços. O segmento pode ser nomeado estaticamente ou você pode configurar o SDK para nomeá-lo dinamicamente com base no cabeçalho do host na solicitação de entrada. A nomeação dinâmica permite agrupar rastreamentos com base no nome de domínio na solicitação e aplicar um nome padrão se o nome não corresponder ao padrão esperado (por exemplo, se o cabeçalho do host for falsificado).

Solicitações encaminhadas

Se um balanceador de carga ou outro intermediário encaminhar uma solicitação para o aplicativo, o X-Ray obterá o IP do cliente do cabeçalho `X-Forwarded-For` na solicitação em vez do IP de origem no pacote IP. O IP do cliente registrado para uma solicitação encaminhada pode ser falsificado; portanto, não é digno de confiança.

Quando uma solicitação é encaminhada, o SDK define um campo adicional no segmento para indicar isso. Se o segmento tiver o campo `x_forwarded_for` definido como `true`, isso significa que o IP do cliente foi obtido no cabeçalho `X-Forwarded-For` na solicitação HTTP.

O manipulador cria um segmento para cada solicitação de entrada com um bloco `http` que contém as seguintes informações:

- Método HTTP: GET, POST, PUT, DELETE etc.
- Endereço do cliente: o endereço IP do cliente que enviou a solicitação.
- Código de resposta: o código de resposta HTTP da solicitação concluída.
- Horário: a hora de início (quando a solicitação foi recebida) e a hora de término (quando a resposta foi enviada).
- Agente do usuário: o `user-agent` da solicitação.
- Tamanho do conteúdo: o `content-length` da resposta.

Configurar uma estratégia de nomeação de segmentos

AWS X-Ray usa um nome de serviço para identificar seu aplicativo e diferenciá-lo dos outros aplicativos, bancos de dados APIs, AWS recursos externos e que seu aplicativo usa. Quando o X-Ray SDK gera segmentos para solicitações recebidas, ele registra o nome do serviço do aplicativo no [campo de nome](#) do segmento.

O X-Ray SDK pode nomear segmentos com o nome do host no cabeçalho da solicitação HTTP. No entanto, esse cabeçalho pode ser falsificado, o que pode resultar em nós inesperados no mapa de serviço. Para evitar que o SDK nomeie segmentos incorretamente devido a solicitações com cabeçalhos de host falsificados, você deve especificar um nome padrão para as solicitações recebidas.

Se o aplicativo atende a solicitações para vários domínios, você pode configurar o SDK para usar uma estratégia de nomeação dinâmica para refletir isso nos nomes dos segmentos. Uma estratégia de nomeação dinâmica permite que o SDK use o nome do host para solicitações que correspondam a um padrão esperado e aplique o nome padrão às solicitações que não correspondem.

Por exemplo, você pode ter um único aplicativo para atender a solicitações para três subdomínios: `www.example.com`, `api.example.com` e `static.example.com`. Você pode usar uma estratégia de nomeação dinâmica com o padrão `*.example.com` a fim de identificar segmentos para cada subdomínio com um nome diferente, o que resulta em três nós de serviço no mapa de serviços. Se o aplicativo receber solicitações com um nome de host que não corresponda ao padrão, você verá um quarto nó no mapa de serviços com um nome alternativo especificado por você.

Para usar o mesmo nome para todos os segmentos da solicitação, especifique o nome de seu aplicativo ao criar o manipulador, conforme mostrado na seção anterior.

Note

Você pode sobrepor o nome do serviço padrão que definiu no código com a [variável de ambiente](#) `AWS_XRAY_TRACING_NAME`.

Uma estratégia de nomeação dinâmica define um padrão com o qual os nomes de host devem corresponder e um nome padrão a ser usado se o nome do host na solicitação HTTP não corresponder ao padrão. Para nomear segmentos dinamicamente, use `NewDynamicSegmentNamer` para configurar o nome padrão e o padrão a ser correspondido.

Example main.go

Se o nome do host na solicitação estiver em conformidade com o padrão `*.example.com`, use o nome do host. Caso contrário, use `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)
}
```

Rastreando chamadas AWS do SDK com o X-Ray SDK for Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Quando seu aplicativo faz chamadas Serviços da AWS para armazenar dados, gravar em uma fila ou enviar notificações, o X-Ray SDK for Go rastreia as chamadas downstream em subsegmentos.

Os recursos rastreados Serviços da AWS e que você acessa nesses serviços (por exemplo, um bucket do Amazon S3 ou uma fila do Amazon SQS) aparecem como nós downstream no mapa de rastreamento no console X-Ray.

Para rastrear clientes do SDK da AWS, encapsule o objeto do cliente com a chamada `xray.AWS()`, conforme mostrado no exemplo a seguir.

Example main.go

```
var dynamo *dynamodb.DynamoDB
func main() {
    dynamo = dynamodb.New(session.Must(session.NewSession()))
    xray.AWS(dynamo.Client)
}
```

Em seguida, quando você usar o cliente do SDK da AWS, use a versão `withContext` do método de chamada e passe-a para `context` no objeto do `http.Request` passado para o [manipulador](#).

Example main.go — chamada SDK AWS

```
func listTablesWithContext(ctx context.Context) {
    output := dynamo.ListTablesWithContext(ctx, &dynamodb.ListTablesInput{})
    doSomething(output)
}
```

Para todos os serviços, o nome da API chamada no console do X-Ray pode ser visto. Para um subconjunto de serviços, o X-Ray SDK adiciona informações ao segmento para fornecer maior detalhamento no mapa de serviços.

Por exemplo, quando você faz uma chamada com um cliente instrumentado do DynamoDB, o SDK adiciona o nome da tabela ao segmento para chamadas direcionadas a uma tabela. No console, cada tabela aparece como um nó separado no mapa de serviços, com um nó genérico do DynamoDB para chamadas não direcionadas a uma tabela.

Example Subsegmento para uma chamada ao DynamoDB para salvar um item

```
{
  "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": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Ao acessar recursos nomeados, as chamadas para os serviços os seguir criam nós adicionais no mapa de serviço. As chamadas que não apontam para recursos específicos criam um nó genérico para o serviço.

- Amazon DynamoDB: nome da tabela
- Amazon Simple Storage Service: nome de chave e bucket
- Amazon Simple Queue Service: nome da fila

Rastrear chamadas para serviços da web HTTP subsequentes com o X-Ray SDK para Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando seu aplicativo faz chamadas para microsserviços ou HTTP público APIs, você pode usar o `xray.Client` para instrumentar essas chamadas como subsegmentos do seu aplicativo Go, conforme mostrado no exemplo a seguir, em que `http-client` é um cliente HTTP.

O cliente cria uma cópia superficial do cliente HTTP fornecido, padronizando para `http.DefaultClient`, com `roundtripper` encapsulado com `xray.RoundTripper`.

Example

<caption>main.go: cliente HTTP</caption>

```
myClient := xray.Client(http-client)
```

<caption>main.go: rastrear uma chamada HTTP subsequente com a biblioteca ctxhttp</caption>

O exemplo a seguir instrumenta a chamada HTTP de saída com a biblioteca `ctxhttp` usando `xray.Client`. `ctx` pode ser transmitido usando a chamada subsequente. Isso garante que o contexto do segmento existente seja usado. Por exemplo, o X-Ray não permite que um novo segmento seja criado em uma função do Lambda; portanto, o contexto existente do segmento do Lambda deve ser usado.

```
resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

Rastrear consultas SQL com o X-Ray SDK para Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Para rastrear chamadas do SQL para PostgreSQL ou MySQL, substituindo as chamadas `sql.Open` por `xray.SQLContext`, conforme mostrado no exemplo a seguir. Use URLs em vez de strings de configuração, se possível.

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  
}
```

Gerar subsegmentos personalizados com o X-Ray SDK para Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Os subsegmentos estendem o [segmento](#) de um rastreamento com detalhes sobre o trabalho realizado para atender a uma solicitação. Sempre que você faz uma chamada com um cliente instrumentado, o X-Ray SDK registra as informações geradas em um subsegmento. Você pode criar subsegmentos adicionais para agrupar outros subsegmentos, medir o desempenho de uma seção do código ou registrar anotações e metadados.

Use o método `Capture` para criar um subsegmento em torno de uma função.

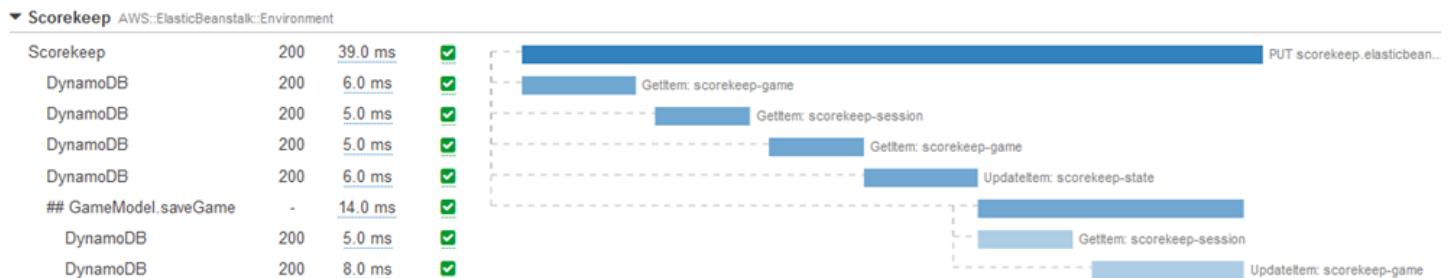
Example main.go: subsegmento personalizado

```
func criticalSection(ctx context.Context) {
    //this is an example of a subsegment
    xray.Capture(ctx, "GameModel.saveGame", func(ctx1 context.Context) error {
        var err error

        section.Lock()
        result := someLockedResource.Go()
        section.Unlock()

        xray.AddMetadata(ctx1, "ResourceResult", result)
    })
}
```

A captura de tela a seguir mostra um exemplo de como o subsegmento `saveGame` pode aparecer em rastreamentos do aplicativo Scorekeep.



Adicionar anotações e metadados aos segmentos com o X-Ray SDK para Go

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais

informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Você pode registrar informações adicionais sobre solicitações, o ambiente ou o aplicativo com anotações e metadados. Também é possível adicionar anotações e metadados aos segmentos que o X-Ray SDK cria ou aos subsegmentos personalizados que você cria.

Anotações são pares de chave-valor com valores booleanos, de string ou número. As anotações são indexadas para serem usadas com [expressões de filtro](#). Use anotações para registrar dados que você deseja usar para agrupar rastreamentos no console ou ao chamar a API [GetTraceSummaries](#).

Metadados são pares chave-valor que podem ter valores de qualquer tipo, incluindo objetos e listas, mas não são indexados para uso com expressões de filtro. Use metadados para registrar dados adicionais que você deseja armazenar no rastreamento e não precisa usar com a pesquisa.

Além de anotações e metadados, você também pode [registrar strings de ID de usuário](#) em segmentos. Os IDs dos usuários são registrados em um campo separado nos segmentos e são indexados para uso com a pesquisa.

Seções

- [Registrar anotações com o X-Ray SDK para Go](#)
- [Registrar metadados com o X-Ray SDK para Go](#)
- [Gravando usuário IDs com o X-Ray SDK for Go](#)

Registrar anotações com o X-Ray SDK para Go

Use anotações para gravar informações sobre segmentos que você deseja indexados para pesquisa.

Requisitos de anotação

- Chaves: a chave para uma anotação do X-Ray pode ter até 500 caracteres alfanuméricos. Você não pode usar espaços ou símbolos que não sejam um ponto (.)
- Valores: o valor de uma anotação do X-Ray pode ter até 1.000 caracteres Unicode.

- O número de anotações: você pode usar até cinquenta anotações por rastreamento.

Para gravar anotações, chame `AddAnnotation` com uma string que contém os metadados que você deseja associar ao segmento.

```
xray.AddAnnotation(key string, value interface{})
```

O SDK registra anotações como pares de chave-valor em um objeto `annotations` no documento de segmentos. Chamar `AddAnnotation` duas vezes com a mesma chave substitui os valores gravados anteriormente no mesmo segmento.

Para encontrar rastreamentos que têm anotações com valores específicos, use a palavra-chave `annotation[key]` em uma [expressão de filtro](#).

Registrar metadados com o X-Ray SDK para Go

Use metadados para gravar informações em segmentos dos quais você não precisa indexados para pesquisa.

Para gravar metadados, chame `AddMetadata` com uma string que contém os metadados que você deseja associar ao segmento.

```
xray.AddMetadata(key string, value interface{})
```

Gravando usuário IDs com o X-Ray SDK for Go

Registre o usuário IDs nos segmentos da solicitação para identificar o usuário que enviou a solicitação.

Para gravar usuário IDs

1. Obtenha uma referência para o segmento atual em `AWSXRay`.

```
import (  
    "context"  
    "github.com/aws/aws-xray-sdk-go/xray"  
)  
  
mySegment := xray.GetSegment(context)
```

2. Chame `setUser` com um ID de string do usuário que enviou a solicitação.

```
mySegment.setUser = "U12345"
```

Para encontrar rastreamentos para um ID de usuário, use a `user` palavra-chave em uma [expressão de filtragem](#).

Como trabalhar com Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Há duas maneiras de instrumentar um aplicativo Java para enviar rastreamentos ao X-Ray:

- [AWS Distro for OpenTelemetry Java](#) — Uma AWS distribuição que fornece um conjunto de bibliotecas de código aberto para enviar métricas e rastreamentos correlacionados para várias soluções de AWS monitoramento, incluindo Amazon e Amazon OpenSearch Service CloudWatch AWS X-Ray, por meio da [AWS Distro](#) for Collector. OpenTelemetry
- [AWS X-Ray SDK for Java — Um conjunto de bibliotecas para](#) gerar e enviar rastreamentos para o X-Ray por meio do daemon [X-Ray](#).

Para obter mais informações, consulte [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#).

AWS Distro para OpenTelemetry Java

Com o AWS Distro para OpenTelemetry (ADOT) Java, você pode instrumentar as aplicações uma vez e enviar métricas e rastreamentos correlacionados para várias soluções de monitoramento da AWS, como o Amazon CloudWatch, o AWS X-Ray e o Amazon OpenSearch Service. O uso do X-Ray com o ADOT requer dois componentes: um SDK do OpenTelemetry habilitado para uso com o X-Ray e o coletor AWS Distro para OpenTelemetry habilitado para uso com o X-Ray. O ADOT Java comporta instrumentação automática, permitindo que a aplicação envie rastreamentos sem alterações no código.

Para começar, consulte a documentação do [AWS Distro para OpenTelemetry Java](#).

Para obter mais informações sobre como usar o AWS Distro para OpenTelemetry com o AWS X-Ray e outros Serviços da AWS, consulte [AWS Distro para OpenTelemetry](#) ou a [documentação do AWS Distro para OpenTelemetry](#).

Para obter mais informações sobre compatibilidade de idiomas e uso, consulte [AWS Observability no GitHub](#).

AWS X-Ray SDK for Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

O X-Ray SDK para Java é um conjunto de bibliotecas para aplicações Java que fornece classes e métodos para gerar e enviar dados de rastreamento ao daemon do X-Ray. Os dados de rastreamento incluem informações sobre solicitações HTTP recebidas pelo aplicativo e chamadas que o aplicativo faz para serviços downstream usando o AWS SDK, clientes HTTP ou um conector de banco de dados SQL. Você também pode criar segmentos manualmente e adicionar informações de depuração em anotações e metadados.

O X-Ray SDK para Java é um projeto de código aberto. Você pode acompanhar o projeto e enviar problemas e pull requests em GitHub: github.com/aws/aws-xray-sdk-java

Comece [adicionando AWSXRayServletFilter como um filtro de servlet](#) para rastrear as solicitações de entrada. Um filtro de servlet cria um [segmento](#). Embora o segmento esteja aberto, você pode usar os métodos do cliente do SDK para adicionar informações ao segmento e criar subsegmentos para rastrear as chamadas subsequentes. O SDK também registra automaticamente exceções que seu aplicativo lança enquanto o segmento está aberto.

A partir da versão 1.3, você pode instrumentar seu aplicativo usando a [programação orientada a aspectos \(AOP\) no Spring](#). Isso significa que você pode instrumentar seu aplicativo enquanto ele está em execução AWS, sem adicionar nenhum código ao tempo de execução do seu aplicativo.

Em seguida, use o X-Ray SDK for Java para instrumentar AWS SDK para Java seus clientes [incluindo o submódulo SDK Instrumentor](#) em sua configuração de compilação. Sempre que você faz uma chamada para um downstream AWS service (Serviço da AWS) ou recurso com um cliente instrumentado, o SDK registra as informações sobre a chamada em um subsegmento. Serviços da AWS e os recursos que você acessa nos serviços aparecem como nós downstream no mapa de rastreamento para ajudá-lo a identificar erros e problemas de limitação em conexões individuais.

Se você não quiser instrumentar todas as chamadas downstream Serviços da AWS, você pode omitir o submódulo Instrumentor e escolher quais clientes instrumentar. Instrumente clientes individuais [adicionando um TracingHandler](#) a a um cliente de serviço do AWS SDK.

Outros submódulos do X-Ray SDK for Java fornecem instrumentação para chamadas downstream para bancos de dados HTTP Web e SQL. APIs Você pode [usar as versões HTTPClient e HTTPClientBuilder](#) do X-Ray SDK para Java no submódulo Apache HTTP para instrumentar clientes do Apache HTTP. Para instrumentar consultas SQL, [adicione o interceptor do SDK para sua fonte de dados](#).

Depois que você começar a usar o SDK, personalize seu comportamento [configurando o gravador e o filtro de servlet](#). Você pode adicionar plug-ins para registrar dados sobre os recursos de computação que executam seu aplicativo, personalizar o comportamento de amostragem, estipulando regras de amostragem, e definir o nível de log para ver mais ou menos informações do SDK nos logs do aplicativo.

Registre informações adicionais sobre as solicitações e o trabalho que o aplicativo faz em [anotações e metadados](#). Anotações são simples pares de chave-valor que são indexados para serem usados com [expressões de filtro](#) para que você possa pesquisar rastreamentos que contêm dados específicos. As entradas de metadados são menos restritivas e podem registrar matrizes e objetos inteiros: tudo o que pode ser serializado em JSON.

Anotações e metadados

Anotações e metadados são textos arbitrários que você adiciona aos segmentos com o X-Ray SDK. As anotações são indexadas para serem usadas com expressões de filtro. Os metadados não são indexados, mas podem ser visualizados no segmento bruto com o console ou a API do X-Ray. Qualquer pessoa à qual você conceder acesso de leitura ao X-Ray poderá visualizar esses dados.

Quando você tem uma grande quantidade de clientes instrumentados em seu código, um único segmento de solicitação pode conter muitos subsegmentos, um para cada chamada feita com um cliente instrumentado. Você pode organizar e agrupar subsegmentos integrando chamadas de clientes em [subsegmentos personalizados](#). Você pode criar um subsegmento personalizado para uma função inteira ou qualquer seção de código e registrar metadados e anotações no subsegmento em vez de gravar tudo no segmento principal.

Submódulos

Você pode baixar o X-Ray SDK para Java do Maven. O X-Ray SDK para Java é dividido em submódulos por caso de uso, com uma lista de materiais para gerenciamento de versões:

- [aws-xray-recorder-sdk-core](#) (obrigatório): funcionalidade básica para a criação e a transmissão de segmentos. Inclui `AWSXRayServletFilter` para instrumentar as solicitações de entrada.
- [aws-xray-recorder-sdk-aws-sdk](#)— Instrumenta as chamadas Serviços da AWS feitas com AWS SDK para Java clientes adicionando um cliente de rastreamento como manipulador de solicitações.
- [aws-xray-recorder-sdk-aws-sdk-v2](#)— Instrumenta as chamadas Serviços da AWS feitas com clientes AWS SDK para Java 2.2 e posteriores adicionando um cliente de rastreamento como um interceptor de solicitações.
- [aws-xray-recorder-sdk-aws-sdk-instrumentor](#)— Com `aws-xray-recorder-sdk-aws-sdk`, instrumenta todos os AWS SDK para Java clientes automaticamente.
- [aws-xray-recorder-sdk-aws-sdk-v2-instrumentor](#)— Com `aws-xray-recorder-sdk-aws-sdk-v2`, instrumenta automaticamente todos os clientes AWS SDK para Java 2.2 e posteriores.
- [aws-xray-recorder-sdk-apache-http](#): instrumenta chamadas HTTP de saída feitas com clientes do Apache HTTP.
- [aws-xray-recorder-sdk-spring](#): fornece interceptadores para aplicações Spring AOP Framework.
- [aws-xray-recorder-sdk-sql-postgres](#): instrumenta chamadas de saída para um banco de dados PostgreSQL feitas com JDBC.
- [aws-xray-recorder-sdk-sql-mysql](#): instrumenta chamadas de saída para um banco de dados MySQL feitas com JDBC.

- [aws-xray-recorder-sdk-bom](#): fornece uma lista de materiais que você pode usar para especificar a versão a ser usada para todos os submódulos.
- [aws-xray-recorder-sdk-metrics](#)— Publique CloudWatch métricas sem amostragem da Amazon a partir de seus segmentos de X-Ray coletados.

Se você usar o Maven ou o Gradle para criar sua aplicação, [adicione o X-Ray SDK para Java à configuração de compilação](#).

Para documentação de referência das classes e métodos do SDK, consulte [Referência de API do AWS X-Ray SDK para Java](#).

Requisitos

O X-Ray SDK for Java requer Java 8 ou posterior, Servlet API 3, AWS SDK e Jackson.

O SDK depende das seguintes bibliotecas para compilação e tempo de execução:

- AWS SDK for Java versão 1.11.398 ou posterior
- API Servlet 3.1.0

Essas dependências são declaradas no arquivo `pom.xml` do SDK e serão incluídas automaticamente se você criar usando Maven ou Gradle.

Se você usar uma biblioteca que esteja incluída no X-Ray SDK para Java, deverá usar a versão incluída. Por exemplo, se você já usa Jackson em tempo de execução e inclui arquivos JAR em sua implantação para essa dependência, deverá remover esses arquivos JAR porque o SDK JAR inclui suas próprias versões de bibliotecas Jackson.

Gerenciamento de dependências

O SDK para Java do X-Ray para Java está disponível no Maven:

- Grupo: `com.amazonaws`
- Artefato: `aws-xray-recorder-sdk-bom`
- Versão: `2.11.0`

Se você usar o Maven para criar seu aplicativo, adicione o SDK como uma dependência no arquivo `pom.xml`.

Example pom.xml: dependências

```
<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>
```

Para Gradle, adicione o SDK como uma dependência de tempo de compilação ao arquivo `build.gradle`.

Example build.gradle: dependências

```
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')
    }
}
```

Se você usar o Elastic Beanstalk para implantar a aplicação, poderá usar o Maven ou o Gradle para criar na instância toda vez que implantar, em vez de criar e carregar um grande arquivo que inclui todas as suas dependências. Consulte o [aplicativo de amostra](#) para obter um exemplo que usa Gradle.

Agente de instrumentação automática do AWS X-Ray para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O agente de AWS X-Ray instrumentação automática para Java é uma solução de rastreamento que instrumenta seus aplicativos web Java com o mínimo esforço de desenvolvimento. O agente permite o rastreamento de aplicativos baseadas em servlets e de todas as solicitações subsequentes do agente feitas com frameworks e bibliotecas compatíveis. Isso inclui solicitações HTTP subsequentes do Apache, solicitações de SDK da AWS e consultas SQL feitas usando um driver JDBC. O agente propaga o contexto do X-Ray, incluindo todos os segmentos e subsegmentos ativos, em todos os threads. A versatilidade e todas as configurações do X-Ray SDK ainda estão disponíveis com o agente do Java. Os padrões adequados foram escolhidos para garantir que o agente trabalhe com o mínimo esforço.

A solução de agente do X-Ray é mais adequada para servidores de aplicativos web Java baseadas em servlets e de solicitação e resposta. Se seu aplicativo usa um framework assíncrono ou não está bem modelada como um serviço de solicitação e resposta, é aconselhável considerar a instrumentação manual com o SDK.

O agente X-Ray é construído usando o kit de ferramentas Distributed Systems Comprehension, ou Di. SCo Di SCo é uma estrutura de código aberto para criar agentes Java que podem ser usados em sistemas distribuídos. Embora não seja necessário entender o Di SCo para usar o agente X-Ray, você pode aprender mais sobre o projeto visitando sua [página inicial em GitHub](#). O agente do X-Ray também é totalmente de código aberto. Para ver o código-fonte, fazer contribuições ou levantar questões sobre o agente, visite seu [repositório em GitHub](#).

Aplicação de exemplo

A aplicação [eb-java-scorekeep](#)da amostra é adaptada para ser instrumentada com o agente X-Ray. Essa ramificação não contém filtro de servlet ou configuração de gravador, pois essas funções são executadas pelo agente. Para executar o aplicativo localmente ou usando recursos da AWS , siga as etapas no arquivo readme do aplicativo de exemplo. As instruções sobre como usar o aplicativo de exemplo para gerar rastreamentos do X-Ray estão no [tutorial do aplicativo de exemplo](#).

Introdução

Para começar a usar o agente do Java de instrumentação automática do X-Ray em seu aplicativo, siga as etapas abaixo.

1. Execute o daemon do X-Ray no seu ambiente. Para obter mais informações, consulte [AWS X-Ray daemon](#).
2. Baixe a [distribuição mais recente do agente](#). Descompacte o arquivo e anote a respectiva localização no sistema de arquivos. O conteúdo deve ser semelhante ao apresentado abaixo.

```
disco
### disco-java-agent.jar
### disco-plugins
### aws-xray-agent-plugin.jar
### disco-java-agent-aws-plugin.jar
### disco-java-agent-sql-plugin.jar
### disco-java-agent-web-plugin.jar
```

3. Modifique os argumentos da JVM do aplicativo para incluir o que vem a seguir, que habilita o agente. O argumento `-javaagent` deve ser colocado antes do argumento `-jar`, se aplicável. O processo para modificar os argumentos da JVM varia de acordo com as ferramentas e os frameworks que você usa para iniciar o servidor Java. Consulte a documentação do framework do servidor para obter orientação específica.

```
-javaagent:/<path-to-disco>/disco-java-agent.jar=pluginPath=/<path-to-disco>/disco-plugins
```

4. Para especificar como o nome do aplicativo aparece no console do X-Ray, defina a variável de ambiente `AWS_XRAY_TRACING_NAME` ou a propriedade do sistema `com.amazonaws.xray.strategy.tracingName`. Se nenhum nome for fornecido, será usado um nome padrão.
5. Reinicie o servidor ou contêiner. As solicitações recebidas e as chamadas subsequentes agora são rastreadas. Se você não vir os resultados esperados, consulte [the section called “Solução de problemas”](#).

Configuração

O agente do X-Ray é configurado por um arquivo JSON externo fornecido pelo usuário. Por padrão, esse arquivo está na raiz do caminho de classe do usuário (por exemplo, no diretório `resources`) chamado `xray-agent.json`. Você pode configurar um local personalizado para o arquivo de configuração definindo a propriedade do sistema `com.amazonaws.xray.configFile` como o caminho absoluto do sistema de arquivos do arquivo de configuração.

Um exemplo de arquivo de configuração é mostrado a seguir.

```
{
  "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
}

```

Especificação de configuração

A tabela a seguir descreve valores válidos para cada propriedade. Os nomes das propriedades diferenciam maiúsculas de minúsculas, mas as chaves não. Para propriedades que podem ser substituídas por variáveis de ambiente e propriedades do sistema, a ordem de prioridade é sempre variável de ambiente, depois propriedade do sistema e, em seguida, arquivo de configuração. Para obter informações sobre as propriedades que você pode substituir, consulte [Variáveis de ambiente](#). Todos os campos são opcionais.

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
serviceName	String	Qualquer string	O nome do serviço instrumentado, conforme ele aparecerá no console do X-Ray.	AWS_XRAY_NOME_DE_FILHO_STREAMING	com.amazonaws.xray.strategy.tracingName	XRayInstrumentedService
contextMissingStrategy	String	LOG_ERROR, LOG_WARN, LOG_INFO, LOG_DEBUG	A ação executada pelo agente de instrumentação automática quando uma exceção é lançada e não há contexto de rastreamento disponível.	AWS_XRAY_CONTEXT_MISSING_STRATEGY	com.amazonaws.xray.strategy.	LOG_ERROR

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
		IGNORE_ERROR	agente quando ele tenta usar o contexto do segmento do X-Ray e não há nenhum presente.		contextMissingStrategy	
daemonAddress	String	Endereço IP e porta formatados ou lista de endereços TCP e UDP	O endereço que o agente usa para se comunicar com o daemon do X-Ray.	AWS_XRAY_ENDPOINT_DAEMON	com.amazonaws.xray.emitter.daemonAddress	127.0.0.1:2000
tracingEnabled	Booleano	True, False	Permite a instrumentação pelo agente do X-Ray.	AWS_XRAY_RASTERIZATION_HABILITADO	com.amazonaws.xray.tracingEnabled	TRUE

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
samplingStrategy	String	CENTRAL, LOCAL, NONE, ALL	A estratégia de amostragem usada pelo agente. ALL captura todas as solicitações, NONE não captura nenhuma solicitação. Consulte as regras de amostragem .	N/D	N/D	CENTRAL
traceInjectionPrefix	String	Qualquer string	Inclui o prefixo fornecido antes do rastreamento injetado IDs nos registros.	N/D	N/D	None (empty string)

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
samplingRulesManifest	String	Um caminho de arquivo absoluto	O caminho para um arquivo de regras de amostragem personalizado a ser usado como fonte de regras de amostragem para a estratégia de amostragem local ou as regras de fallback para a estratégia central.	N/D	N/D	DefaultSamplingRules.json

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
awsServiceHandlerManifest	String	Um caminho de arquivo absoluto	O caminho para uma lista de permissões de parâmetros personalizados, o qual captura informações adicionais de clientes de SDK da AWS .	N/D	N/D	DefaultOperationParameterWhitelist.json
awsSdkVersion	Inteiro	1, 2	Versão do AWS SDK para Java que você está usando. Ignorado se <code>awsServiceHandlerManifest</code> também não estiver definido.	N/D	N/D	2

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
maxStackTraceComprimeto	Inteiro	Números inteiros não negativos	O máximo de linhas de um rastreamento de pilha a serem registradas em um rastreamento.	N/D	N/D	50
streamingThreshold	Inteiro	Números inteiros não negativos	Depois que pelo menos tantos subsegmentos são fechados, eles são transmitidos para o daemon out-of-band para evitar que os pedaços sejam muito grandes.	N/D	N/D	100

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
tracedInjection	Booleano	True, False	Permite a injeção de ID de rastreamento do X-Ray nos logs se as dependências e a configuração descritas na configuração de registro em log também forem adicionadas. Caso contrário, não faz nada.	N/D	N/D	TRUE

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
pluginsEnabled	Booleano	True, False	Ativa plugins que registram metadados sobre os AWS ambientes em que você está operando. Consulte Plug-ins .	N/D	N/D	TRUE
collectSqlQueries	Booleano	True, False	Registra strings de consulta SQL em subsegmentos de SQL com base no melhor esforço.	N/D	N/D	FALSE

Nome da propriedade	Tipo	Valores válidos	Description	Variável de ambiente	Propriedades do sistema	Padrão
contextPropagation	Booleano	True, False	Se verdadeiro, propaga automaticamente o contexto do X-Ray entre os segmentos. Caso contrário, usa Thread Local para armazenar contexto, e a propagação manual entre threads é necessária.	N/D	N/D	TRUE

Configuração de registro

O nível de log do agente do X-Ray pode ser configurado da mesma forma que o X-Ray SDK para Java. Consulte [Registro em log](#) para obter mais informações sobre como configurar o registro em log com o X-Ray SDK para Java.

Instrumentação manual

Se você quiser realizar instrumentação manual além da instrumentação automática do agente, adicione o X-Ray SDK como uma dependência ao seu projeto. Observe que os filtros de servlet

personalizados do SDK mencionados em [Rastrear solicitações de entrada](#) não são compatíveis com o agente do X-Ray.

Note

Você deve usar a versão mais recente do X-Ray SDK para realizar a instrumentação manual e, ao mesmo tempo, usar o agente.

Se você estiver trabalhando em um projeto Maven, adicione as dependências a seguir ao arquivo `pom.xml`.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
    <version>2.11.0</version>
  </dependency>
</dependencies>
```

Se você estiver trabalhando em um projeto Gradle, adicione as dependências a seguir ao arquivo `build.gradle`.

```
implementation 'com.amazonaws:aws-xray-recorder-sdk-core:2.11.0'
```

Você pode adicionar [subsegmentos personalizados](#), além de [anotações, metadados e usuário IDs](#) enquanto usa o agente, assim como faria com o SDK normal. Como o agente propaga automaticamente o contexto entre os threads, nenhuma solução alternativa para propagar o contexto deve ser necessária ao trabalhar com aplicativos de vários threads.

Solução de problemas

Como o agente oferece instrumentação totalmente automática, pode ser difícil identificar a causa raiz quando você está enfrentando problemas. Se o agente do X-Ray não estiver funcionando conforme o esperado, analise os problemas e soluções a seguir. O agente do X-Ray e o respectivo SDK usam o Jakarta Commons Logging (JCL). Para ver a saída do registro em log, uma ponte para conectar o JCL ao back-end de registro em log deve estar no caminho de classe, como no exemplo a seguir: `log4j-jcl` ou `jcl-over-slf4j`.

Problema: Eu habilitei o agente do Java no aplicativo, mas não vejo nada no console X-Ray

O daemon do X-Ray está sendo executado na mesma máquina?

Caso contrário, consulte a [documentação do daemon do X-Ray](#) para configurá-lo.

Nos logs do aplicativo, você vê uma mensagem como “Inicializando o gravador do agente do X-Ray”?

Se você adicionou corretamente o agente ao aplicativo, essa mensagem será registrada em log no nível INFO quando o aplicativo for iniciada, antes que as solicitações comecem a ser recebidas. Se essa mensagem não estiver ali, isso significa que o agente do Java não está sendo executado com seu processo Java. Confirme se você seguiu todas as etapas de configuração corretamente, sem erros de digitação.

Nos registros do seu aplicativo, você vê várias mensagens de erro dizendo algo como “Suprimindo a exceção ausente do AWS X-Ray contexto”?

Esses erros ocorrem porque o agente está tentando instrumentar solicitações downstream, como solicitações de AWS SDK ou consultas SQL, mas não conseguiu criar automaticamente um segmento. Se você observar muitos erros desse tipo, talvez o agente não seja a melhor ferramenta para seu caso de uso. Em vez disso, é aconselhável considerar a instrumentação manual com o X-Ray SDK. Como alternativa, você pode habilitar os [logs de depuração](#) do X-Ray SDK para ver o rastreamento da pilha em que as exceções de contexto ausente estão ocorrendo. Você pode agrupar essas partes do código com segmentos personalizados, o que deve resolver esses erros. Para ver um exemplo de empacotamento de solicitações subsequentes com segmentos personalizados, consulte o código de exemplo em [Instrumentar código de inicialização](#).

Problema: Alguns dos segmentos que eu espero não aparecem no console do X-Ray

Suo aplicativo usa vários threads?

Se alguns segmentos que você espera que sejam criados não estiverem aparecendo no console, os threads em segundo plano no aplicativo podem ser a causa. Se seu aplicativo executa tarefas usando encadeamentos em segundo plano que são “acionar e esquecer”, como fazer uma chamada única para uma função Lambda com AWS o SDK ou pesquisar periodicamente algum endpoint HTTP, isso pode confundir o agente enquanto ele propaga o contexto entre os encadeamentos. Para verificar se esse é o seu problema, habilite os logs de depuração do X-Ray SDK e verifique se há mensagens como: Não emitindo segmento chamado <NOME> porque ele gera subsegmentos

em andamento. Para contornar isso, você pode tentar unir os threads em segundo plano antes que o servidor retorne para garantir que todo o trabalho realizado neles seja registrado. Ou você pode definir a configuração `contextPropagation` do agente como `false` para desabilitar a propagação de contexto em threads em segundo plano. Se você fizer isso, precisará instrumentar manualmente esses threads com segmentos personalizados ou ignorar as exceções de contexto ausente que eles produzem.

Você configurou regras de amostragem?

Se segmentos aparentemente aleatórios ou inesperados estiverem aparecendo no console do X-Ray, ou os segmentos que você espera que estejam no console não estiverem aparecendo, você pode estar enfrentando um problema de amostragem. O agente do X-Ray aplica amostragem centralizada a todos os segmentos que ele cria, usando as regras do console do X-Ray. A regra de amostragem padrão é 1 segmento por segundo, mais 5% dos segmentos posteriores. Isso significa que os segmentos criados rapidamente com o agente podem não ser amostrados. Para resolver isso, você deve criar regras de amostragem personalizadas no console do X-Ray que tirem amostras adequadas dos segmentos desejados. Para obter mais informações, consulte [Amostragem](#).

Configurar o X-Ray SDK para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O X-Ray SDK para Java tem uma classe chamada `AWSXRay`, que fornece o gravador global. Este é um `TracingHandler` que pode ser usado para instrumentar o código. Você pode configurar o gravador global para personalizar o `AWSXRayServletFilter` que cria segmentos para chamadas HTTP de entrada.

Seções

- [Plug-ins de serviço](#)

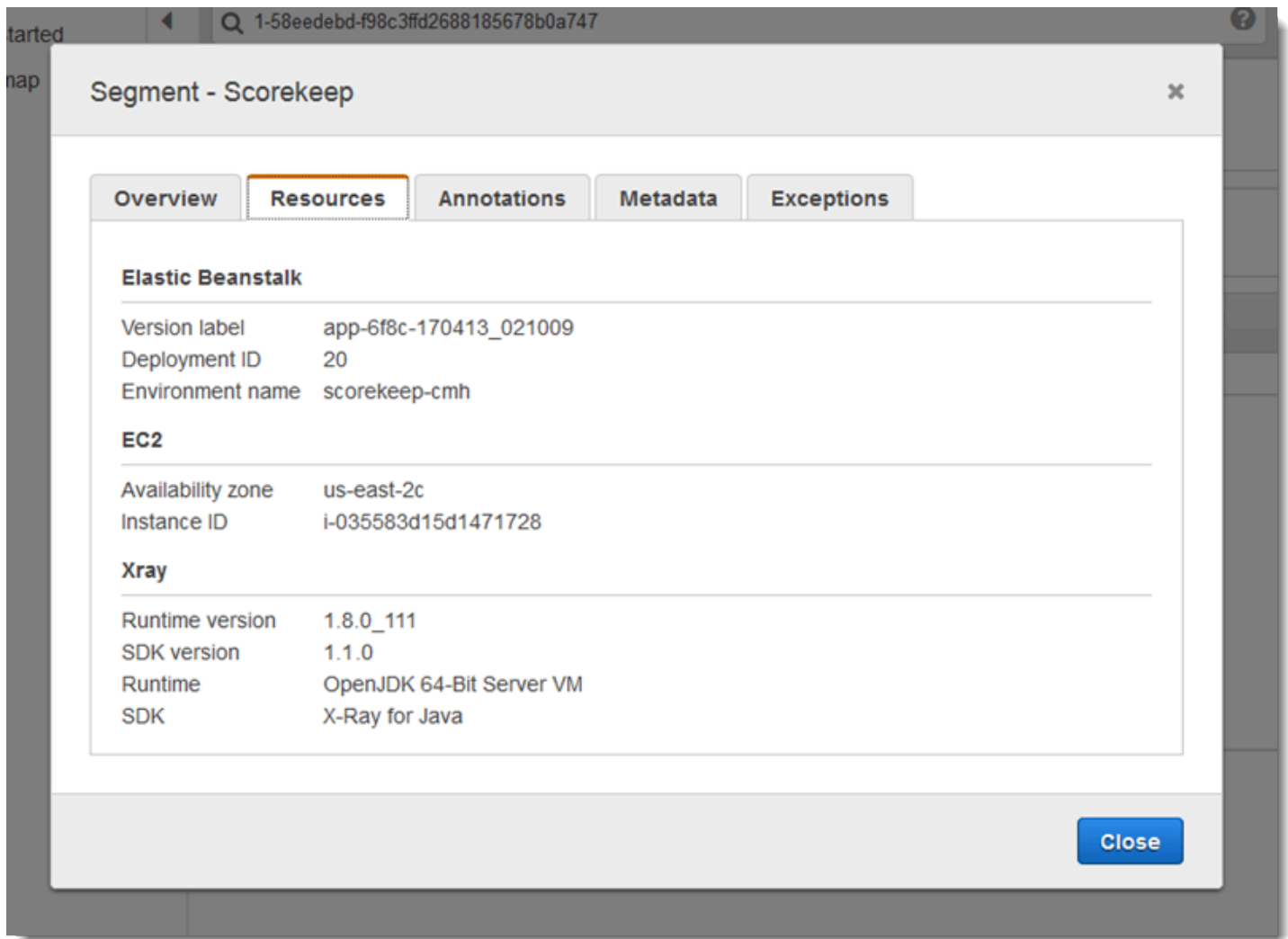
- [Regras de amostragem](#)
- [Registro em log](#)
- [Listeners de segmento](#)
- [Variáveis de ambiente](#)
- [Propriedades do sistema](#)

Plug-ins de serviço

Use plugins para registrar informações sobre o serviço que hospeda o aplicativo.

Plugins

- Amazon EC2 — EC2Plugin adiciona o ID da instância, a zona de disponibilidade e o grupo de CloudWatch registros.
- Elastic Beanstalk: o ElasticBeanstalkPlugin adiciona o nome do ambiente, o rótulo da versão e o ID de implantação.
- Amazon ECS: o ECSPPlugin adiciona o ID do contêiner.
- Amazon EKS — EKSPPlugin adiciona o ID do contêiner, o nome do cluster, o ID do pod e o grupo de CloudWatch registros.



Para usar um plugin, acione o `withPlugin` em seu `AWSXRayRecorderBuilder`.

Example `src/main/java/scorekeep/WebConfig.java` - gravador

```
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());
}
}
```

O SDK também usa as configurações do plug-in para definir o campo `origin` no segmento. Isso indica o tipo de AWS recurso que executa seu aplicativo. Quando você usa vários plug-ins, o SDK usa a seguinte ordem de resolução para determinar a origem: ElasticBeanstalk > EKS > ECS >. EC2

Regras de amostragem

O SDK usa as regras de amostragem que você define no console do X-Ray para determinar quais solicitações serão registradas. A regra padrão rastreia a primeira solicitação a cada segundo e 5% de todas as solicitações adicionais em todos os serviços que enviam rastreamentos ao X-Ray. [Crie regras adicionais no console do X-Ray](#) para personalizar a quantidade de dados registrados para cada uma dos aplicativos.

O SDK aplica regras personalizadas na ordem em que elas estão definidas. Se uma solicitação corresponder a várias regras personalizadas, o SDK aplicará somente a primeira regra.

Note

Se o SDK não conseguir acessar o X-Ray para obter regras de amostragem, ele reverterá para uma regra local padrão da primeira solicitação a cada segundo e 5% de todas as solicitações adicionais por host. Isso pode ocorrer se o host não tiver permissão para chamar a amostragem APIs ou não conseguir se conectar ao daemon X-Ray, que atua como um proxy TCP para chamadas de API feitas pelo SDK.

Você também pode configurar o SDK para carregar regras de amostragem de um documento JSON. O SDK pode usar regras locais como backup para casos em que a amostragem do X-Ray não está disponível ou usar exclusivamente regras locais.

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
}
}
```

Este exemplo define uma regra personalizada e uma regra padrão. A regra personalizada aplica uma taxa de amostragem de 5% sem um número mínimo de solicitações para rastrear os caminhos em `/api/move/`. A regra padrão rastreia a primeira solicitação a cada segundo e 10% das solicitações adicionais.

A desvantagem de definir regras localmente é que o destino fixo é aplicado por instância do gravador de forma independente, em vez de ser gerenciado pelo serviço X-Ray. À medida que você implanta mais hosts, a taxa fixa é multiplicada, dificultando o controle da quantidade de dados registrados.

Ativado AWS Lambda, você não pode modificar a taxa de amostragem. Se sua função for chamada por um serviço instrumentado, as chamadas que geraram solicitações que foram amostradas por esse serviço serão registradas pelo Lambda. Se o rastreamento ativo estiver habilitado e nenhum cabeçalho de rastreamento estiver presente, o Lambda tomará a decisão de amostragem.

Para fornecer regras de backup em Spring, configure o gravador global com uma `CentralizedSamplingStrategy` em uma classe de configuração.

Example `src/main/java/myapp/WebConfig.java` - configuração do gravador

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;
```

```
@Configuration
public class WebConfig {

    static {
        AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
        EC2Plugin());

        URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
        builder.withSamplingStrategy(new CentralizedSamplingStrategy(ruleFile));

        AWSXRay.setGlobalRecorder(builder.build());
    }
}
```

Para Tomcat, adicione um listener que estenda `ServletContextListener` e registre o listener na descrição da implantação.

Example `src/com/myapp/web/Startup.java`

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

import java.net.URL;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class Startup implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent event) {
        AWSXRayRecorderBuilder builder =
        AWSXRayRecorderBuilder.standard().withPlugin(new EC2Plugin());

        URL ruleFile = Startup.class.getResource("/sampling-rules.json");
        builder.withSamplingStrategy(new CentralizedSamplingStrategy(ruleFile));

        AWSXRay.setGlobalRecorder(builder.build());
    }

    @Override
    public void contextDestroyed(ServletContextEvent event) { }
}
```


Example WEB-INF/web.xml

```
...  
<listener>  
  <listener-class>com.myapp.web.Startup</listener-class>  
</listener>
```

Para usar somente regras locais, substitua o `CentralizedSamplingStrategy` por uma `LocalizedSamplingStrategy`.

```
builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));
```

Registro em log

Por padrão, o SDK envia mensagens de nível de `ERROR` da aplicação para os logs da aplicação. Você pode habilitar o registro em log em nível de depuração no SDK para gerar logs mais detalhados no arquivo de log da aplicação. Os níveis de log válidos são `DEBUG`, `INFO`, `WARN`, `ERROR` e `FATAL`. O nível de log `FATAL` silencia todas as mensagens de log porque o SDK não registra em log no nível fatal.

Example application.properties

Defina o nível de registro com a propriedade `logging.level.com.amazonaws.xray`.

```
logging.level.com.amazonaws.xray = DEBUG
```

Use logs de depuração para identificar problemas como subsegmentos não fechados ao [gerar subsegmentos manualmente](#).

Injeção de ID de rastreamentos em logs

Para expor o ID de rastreamento totalmente qualificado atual às instruções de log, é possível injetar o ID no contexto de diagnóstico mapeado (MDC). Usando a interface `SegmentListener`, os métodos são chamados do gravador do X-Ray durante eventos do ciclo de vida do segmento. Quando um segmento ou subsegmento começa, o ID de rastreamento qualificado é injetado no MDC com a chave `AWS-XRAY-TRACE-ID`. Quando esse segmento termina, a chave é removida do MDC. Isso expõe o ID de rastreamento à biblioteca de log em uso. Quando um subsegmento termina, seu ID pai é injetado no MDC.

Example ID de rastreamento totalmente qualificado

O ID totalmente qualificado é representado como `TraceID@EntityID`

```
1-5df42873-011e96598b447dfca814c156@541b3365be3dafc3
```

Esse recurso funciona com aplicativos Java instrumentados com o AWS X-Ray SDK for Java e oferece suporte às seguintes configurações de registro:

- SLF4API de front-end J com back-end Logback
- SLF4API de front-end J com back-end Log4J2
- API front-end Log4J2 com backend Log4J2

Consulte as guias a seguir para obter as necessidades de cada front-end e de cada backend.

SLF4J Frontend

1. Adicione a seguinte dependência Maven ao seu projeto.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-xray-recorder-sdk-slf4j</artifactId>
  <version>2.11.0</version>
</dependency>
```

2. Inclua o método `withSegmentListener` ao construir o `AWSXRayRecorder`. Isso adiciona uma `SegmentListener` classe, que injeta automaticamente um novo traço IDs no SLF4 J MDC.

O `SegmentListener` usa uma string opcional como um parâmetro para configurar o prefixo da instrução de log. O prefixo pode ser configurado das seguintes maneiras:

- Nenhum: usa o prefixo `AWS-XRAY-TRACE-ID` padrão.
- Vazio: usa uma string vazia (por exemplo, `""`).
- Personalizado: usa um prefixo personalizado conforme definido na string.

Example Instrução **AWSXRayRecorderBuilder**

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
```

```
.standard().withSegmentListener(new SLF4JSegmentListener("CUSTOM-PREFIX"));
```

Log4J2 front end

1. Adicione a seguinte dependência Maven ao seu projeto.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-xray-recorder-sdk-log4j</artifactId>
  <version>2.11.0</version>
</dependency>
```

2. Inclua o método `withSegmentListener` ao construir o `AWSXRayRecorder`. Isso adicionará uma `SegmentListener` classe, que injeta automaticamente um novo rastreamento totalmente qualificado IDs no SLF4 J MDC.

O `SegmentListener` usa uma string opcional como um parâmetro para configurar o prefixo da instrução de log. O prefixo pode ser configurado das seguintes maneiras:

- Nenhum: usa o prefixo `AWS-XRAY-TRACE-ID` padrão.
- Vazio: usa uma string vazia (por exemplo, `""`) e remove o prefixo.
- Personalizado: usa o prefixo personalizado definido na string.

Example Instrução `AWSXRayRecorderBuilder`

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
    .standard().withSegmentListener(new Log4JSegmentListener("CUSTOM-
    PREFIX"));
```

Logback backend

Para inserir o ID de rastreamento em seus eventos de log, você deve modificar o `PatternLayout` do registrador de logs, que formata cada instrução de log.

1. Localize onde o `patternLayout` está configurado. Isto pode ser feito programaticamente ou através de um arquivo de configuração XML. Para saber mais, consulte [Configuração de Logback](#).

2. Insira `%X{AWS-XRAY-TRACE-ID}` em qualquer lugar no `patternLayout` para inserir o ID de rastreamento em instruções de log futuras. O `%X{}` indica que você está recuperando um valor com a chave fornecida do MDC. Para saber mais sobre o `PatternLayouts Logback`, consulte [PatternLayout](#).

Log4J2 backend

1. Localize onde o `patternLayout` está configurado. Isto pode ser feito programaticamente ou através de um arquivo de configuração escrito no formato XML, JSON, YAML ou de propriedades.

Para saber mais sobre como configurar o Log4J2 por meio de um arquivo de configuração, consulte [Configuração](#).

Para saber mais sobre como configurar o Log4J2 programaticamente, consulte [Configuração programática](#).

2. Insira `%X{AWS-XRAY-TRACE-ID}` em qualquer lugar no `PatternLayout` para inserir o ID de rastreamento em instruções de log futuras. O `%X{}` indica que você está recuperando um valor com a chave fornecida do MDC. [Para saber mais sobre o PatternLayouts Log4J2, consulte Pattern Layout](#).

Exemplo de injeção de ID de rastreamento

A seguir, é mostrado uma string `PatternLayout` modificada para incluir o ID de rastreamento. O ID de rastreamento é impresso após o nome do thread (`%t`) e antes do nível de log (`%-5p`).

Exemplo **PatternLayout** com injeção de ID

```
%d{HH:mm:ss.SSS} [%t] %X{AWS-XRAY-TRACE-ID} %-5p %m%n
```

AWS X-Ray imprime automaticamente a chave e o ID de rastreamento na instrução de log para facilitar a análise. A seguir é mostrada uma instrução de log usando o `PatternLayout` modificado.

Exemplo Instrução de log com injeção de ID

```
2019-09-10 18:58:30.844 [nio-5000-exec-4] AWS-XRAY-TRACE-ID:
1-5d77f256-19f12e4eaa02e3f76c78f46a@1ce7df03252d99e1 WARN 1 - Your logging message
here
```

A mensagem de log em si é alojada no %m padrão e definida ao chamar o registrador de log.

Listeners de segmento

Os receptores de segmento são uma interface para interceptar eventos de ciclo de vida, como o início e o fim de segmentos produzidos pelo `AWSXRayRecorder`. A implementação de uma função de evento do receptor de segmento pode consistir em adicionar a mesma anotação a todos os subsegmentos quando eles são criados com [onBeginSubsegment](#), registrar em log uma mensagem após cada segmento ser enviado para o daemon usando [afterEndSegment](#) ou registrar consultas enviadas pelos interceptores SQL usando [beforeEndSubsegment](#) para verificar se o subsegmento representa uma consulta SQL, adicionando outros metadados em caso afirmativo.

Para ver a lista completa de funções `SegmentListener`, acesse a documentação do [AWS X-Ray X-Ray Recorder SDK para Java API](#).

O exemplo a seguir mostra como adicionar uma anotação consistente a todos os subsegmentos na criação com [onBeginSubsegment](#) e imprimir uma mensagem de log no final de cada segmento com [afterEndSegment](#).

Example MySegmentListener.java

```
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
import com.amazonaws.xray.listeners.SegmentListener;

public class MySegmentListener implements SegmentListener {
    .....

    @Override
    public void onBeginSubsegment(Subsegment subsegment) {
        subsegment.putAnnotation("annotationKey", "annotationValue");
    }

    @Override
    public void afterEndSegment(Segment segment) {
        // Be mindful not to mutate the segment
        logger.info("Segment with ID " + segment.getId());
    }
}
```

Esse listener de segmento personalizado é então referenciado ao criar o `AWSXRayRecorder`.

Example AWSXRayRecorderBuilder declaração

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
    .standard().withSegmentListener(new MySegmentListener());
```

Variáveis de ambiente

Você pode usar variáveis de ambiente para configurar o X-Ray SDK para Java. O SDK é compatível com as variáveis a seguir.

- `AWS_XRAY_CONTEXT_MISSING`: defina como `RUNTIME_ERROR` para lançar exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.

Valores válidos

- `RUNTIME_ERROR`: lance uma exceção de tempo de execução.
- `LOG_ERROR`: registre um erro e continue (padrão).
- `IGNORE_ERROR`: ignore o erro e continue.

Erros relativos a segmentos ou subsegmentos ausentes poderão ocorrer quando você tentar usar um cliente instrumentado no código de inicialização que é executado quando nenhuma solicitação estiver aberta ou em um código que gere um novo thread.

- `AWS_XRAY_DAEMON_ADDRESS`: defina o host e a porta do receptor do daemon do X-Ray. Por padrão, o SDK usa `127.0.0.1:2000` para dados de rastreamento (UDP) e para amostragem (TCP). Use essa variável se você tiver configurado o daemon para [escutar em uma porta diferente](#) ou se ele estiver sendo executado em um host diferente.

Formato

- Mesma porta: `address:port`
- Portas diferentes: `tcp:address:port` `udp:address:port`
- `AWS_LOG_GROUP`: defina o nome do grupo de logs associado ao aplicativo. Se o seu grupo de registros usar a mesma AWS conta e região do seu aplicativo, o X-Ray pesquisará automaticamente os dados de segmento do seu aplicativo usando esse grupo de registros especificado. Para obter mais informações sobre os grupos de logs, consulte [Trabalhar com grupos de logs e fluxos](#).

- `AWS_XRAY_TRACING_NAME`: defina um nome de serviço para o SDK usar para segmentos. Sobreponha o nome do serviço que você definiu na [estratégia de nomeação de segmentos](#) do filtro do servlet.

As variáveis de ambiente substituem as [propriedades do sistema](#) equivalentes e os valores definidos no código.

Propriedades do sistema

É possível usar propriedades do sistema como uma alternativa específica de JVM a [variáveis de ambiente](#). O SDK é compatível com as propriedades a seguir.

- `com.amazonaws.xray.strategy.tracingName`: equivalente ao `AWS_XRAY_TRACING_NAME`.
- `com.amazonaws.xray.emitters.daemonAddress`: equivalente ao `AWS_XRAY_DAEMON_ADDRESS`.
- `com.amazonaws.xray.strategy.contextMissingStrategy`: equivalente ao `AWS_XRAY_CONTEXT_MISSING`.

Caso uma propriedade do sistema e a variável de ambiente equivalente sejam definidas, são usados os valores da variável de ambiente. Ambos os métodos substituem valores definidos no código.

Rastrear solicitações recebidas com o X-Ray SDK para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode usar o X-Ray SDK para rastrear solicitações HTTP recebidas que seu aplicativo atende em uma EC2 instância na Amazon EC2 ou no Amazon AWS Elastic Beanstalk ECS.

Use um `Filter` para instrumentar solicitações HTTP de entrada. Quando você adiciona o filtro de servlet do X-Ray ao aplicativo, o X-Ray SDK para Java cria um segmento para cada solicitação amostrada. Este segmento inclui tempo, método e disposição da solicitação HTTP. Instrumentação adicional cria subsegmentos sobre esse segmento.

Note

Para AWS Lambda funções, o Lambda cria um segmento para cada solicitação de amostra. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Cada segmento tem um nome que identifica o aplicativo no mapa de serviços. O segmento pode ser nomeado estaticamente ou você pode configurar o SDK para nomeá-lo dinamicamente com base no cabeçalho do host na solicitação de entrada. A nomeação dinâmica permite agrupar rastreamentos com base no nome de domínio na solicitação e aplicar um nome padrão se o nome não corresponder ao padrão esperado (por exemplo, se o cabeçalho do host for falsificado).

Solicitações encaminhadas

Se um balanceador de carga ou outro intermediário encaminhar uma solicitação para o aplicativo, o X-Ray obterá o IP do cliente do cabeçalho `X-Forwarded-For` na solicitação em vez do IP de origem no pacote IP. O IP do cliente registrado para uma solicitação encaminhada pode ser falsificado; portanto, não é digno de confiança.

Quando uma solicitação é encaminhada, o SDK define um campo adicional no segmento para indicar isso. Se o segmento tiver o campo `x_forwarded_for` definido como `true`, isso significa que o IP do cliente foi obtido no cabeçalho `X-Forwarded-For` na solicitação HTTP.

O manipulador de mensagens cria um segmento para cada solicitação recebida com um bloco `http` que contém as seguintes informações:

- Método HTTP: GET, POST, PUT, DELETE etc.
- Endereço do cliente: o endereço IP do cliente que enviou a solicitação.
- Código de resposta: o código de resposta HTTP da solicitação concluída.
- Horário: a hora de início (quando a solicitação foi recebida) e a hora de término (quando a resposta foi enviada).
- Agente do usuário: o `user-agent` da solicitação.

- Tamanho do conteúdo: o `content-length` da resposta.

Seções

- [Adicionar um filtro de rastreamento ao aplicativo \(Tomcat\)](#)
- [Adicionar um filtro de rastreamento ao aplicativo \(spring\)](#)
- [Configurar uma estratégia de nomeação de segmentos](#)

Adicionar um filtro de rastreamento ao aplicativo (Tomcat)

Para Tomcat, adicione um `<filter>` ao arquivo `web.xml` do seu projeto. Use o parâmetro `fixedName` para especificar um [nome de serviço](#) para aplicar a segmentos criados para solicitações recebidas.

Example WEB-INF/web.xml: Tomcat

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.java.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>
```

Adicionar um filtro de rastreamento ao aplicativo (spring)

Para Spring, adicione um `Filter` à classe `WebConfig`. Passe o nome do segmento para o construtor [AWSXRayServletFilter](#) como uma string.

Example `src/main/java/myapp/WebConfig.java` - primavera

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
```

```
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;  
  
@Configuration  
public class WebConfig {  
  
    @Bean  
    public Filter TracingFilter() {  
        return new AWSXRayServletFilter("Scorekeep");  
    }  
}
```

Configurar uma estratégia de nomeação de segmentos

AWS X-Ray usa um nome de serviço para identificar seu aplicativo e diferenciá-lo dos outros aplicativos, bancos de dados APIs, AWS recursos externos e que seu aplicativo usa. Quando o X-Ray SDK gera segmentos para solicitações recebidas, ele registra o nome do serviço do aplicativo no [campo de nome](#) do segmento.

O X-Ray SDK pode nomear segmentos com o nome do host no cabeçalho da solicitação HTTP. No entanto, esse cabeçalho pode ser falsificado, o que pode resultar em nós inesperados no mapa de serviço. Para evitar que o SDK nomeie segmentos incorretamente devido a solicitações com cabeçalhos de host falsificados, você deve especificar um nome padrão para as solicitações recebidas.

Se o aplicativo atende a solicitações para vários domínios, você pode configurar o SDK para usar uma estratégia de nomeação dinâmica para refletir isso nos nomes dos segmentos. Uma estratégia de nomeação dinâmica permite que o SDK use o nome do host para solicitações que correspondam a um padrão esperado e aplique o nome padrão às solicitações que não correspondem.

Por exemplo, você pode ter um único aplicativo para atender a solicitações para três subdomínios: `www.example.com`, `api.example.com` e `static.example.com`. Você pode usar uma estratégia de nomeação dinâmica com o padrão `*.example.com` a fim de identificar segmentos para cada subdomínio com um nome diferente, o que resulta em três nós de serviço no mapa de serviços. Se a aplicação receber solicitações com um nome de host que não corresponda ao padrão, você verá um quarto nó no mapa de serviços com um nome alternativo especificado por você.

Para usar o mesmo nome para todos os segmentos de solicitação, especifique o nome do seu aplicativo ao inicializar o filtro de servlet, como mostrado na [seção anterior](#). Isso tem o mesmo efeito que criar uma correção [SegmentNamingStrategy](#) chamando-a `SegmentNamingStrategy.fixed()` e passando-a para o [AWSXRayServletFilter](#) construtor.

 Note

Você pode sobrepor o nome do serviço padrão que definiu no código com a [variável de ambiente](#) `AWS_XRAY_TRACING_NAME`.

Uma estratégia de nomenclatura dinâmica define um padrão que os nomes de host devem corresponder, e um nome padrão a ser usado se o nome do host na solicitação HTTP não corresponder ao padrão. Para nomear segmentos dinamicamente no Tomcat, use o `dynamicNamingRecognizedHosts` e o `dynamicNamingFallbackName` para definir o padrão e o nome padrão, respectivamente.

Example WEB-INF/web.xml – filtro de servlet com nomeação dinâmica

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.javax.servlet.AWSXRayServletFilter</filter-class>
  <init-param>
    <param-name>dynamicNamingRecognizedHosts</param-name>
    <param-value>*.example.com</param-value>
  </init-param>
  <init-param>
    <param-name>dynamicNamingFallbackName</param-name>
    <param-value>MyApp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>AWSXRayServletFilter</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
```

Para Spring, crie uma dinâmica

[SegmentNamingStrategy](#) `SegmentNamingStrategy.dynamic()` chamando e passe-a para o `AWSXRayServletFilter` construtor.

Example src/main/java/myapp/WebConfig.java - filtro de servlet com nomenclatura dinâmica

```
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\("MyApp",  
        "\*.example.com"\));  
    }  
}
```

Rastreando chamadas AWS do SDK com o X-Ray SDK for Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Quando seu aplicativo faz chamadas Serviços da AWS para armazenar dados, gravar em uma fila ou enviar notificações, o X-Ray SDK for Java rastreia as chamadas downstream em subsegmentos.

Os Serviços da AWS rastreados e os recursos que você acessa nesses serviços (por exemplo, um bucket do Amazon S3 ou uma fila do Amazon SQS) são exibidos como nós subsequentes no mapa de serviço no console do X-Ray.

O X-Ray SDK para Java instrumenta automaticamente todos os clientes de SDK da AWS quando você inclui o `aws-sdk` e [submódulos](#) `aws-sdk-instrumentor` na compilação. Se você não incluir o submódulo `Instrumentor`, poderá optar por instrumentar alguns clientes e, ao mesmo tempo, excluir outros.

Para instrumentar clientes individuais, remova o `aws-sdk-instrumentor` submódulo da sua compilação e adicione um `XRayClient` como `TracingHandler` em seu cliente AWS SDK usando o construtor de clientes do serviço.

Por exemplo, para instrumentar um cliente AmazonDynamoDB, transmita um manipulador de rastreamento para `AmazonDynamoDBClientBuilder`.

Example MyModel.java - cliente DynamoDB

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.handlers.TracingHandler;

...
public class MyModel {
    private AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
        .withRegion(Regions.fromName(System.getenv("AWS_REGION")))
        .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder\(\)))
        .build();
    ...
}
```

Para todos os serviços, o nome da API chamada no console do X-Ray pode ser visto. Para um subconjunto de serviços, o X-Ray SDK adiciona informações ao segmento para fornecer maior detalhamento no mapa de serviços.

Por exemplo, quando você faz uma chamada com um cliente instrumentado do DynamoDB, o SDK adiciona o nome da tabela ao segmento para chamadas direcionadas a uma tabela. No console, cada tabela aparece como um nó separado no mapa de serviços, com um nó genérico do DynamoDB para chamadas não direcionadas a uma tabela.

Example Subsegmento para uma chamada ao DynamoDB para salvar um item

```
{
  "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": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",  
  }  
}
```

Ao acessar recursos nomeados, as chamadas para os serviços os seguir criam nós adicionais no mapa de serviço. As chamadas que não apontam para recursos específicos criam um nó genérico para o serviço.

- Amazon DynamoDB: nome da tabela
- Amazon Simple Storage Service: nome de chave e bucket
- Amazon Simple Queue Service: nome da fila

Para instrumentar chamadas downstream para Serviços da AWS com AWS SDK para Java 2.2 e versões posteriores, você pode omitir o `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` módulo da sua configuração de compilação. Inclua o `aws-xray-recorder-sdk-aws-sdk-v2` module em seu lugar e instrumente clientes individuais, configurando-os com um `TracingInterceptor`.

Example AWS SDK para Java 2.2 e posterior - interceptor de rastreamento

```
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();  
    //...  
}
```

Rastrear chamadas para serviços da web HTTP subsequentes com o X-Ray SDK para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando seu aplicativo faz chamadas para microsserviços ou HTTP público APIs, você pode usar a versão do X-Ray SDK for Java `HttpClient` para instrumentar essas chamadas e adicionar a API ao gráfico de serviços como um serviço downstream.

O X-Ray SDK for Java `DefaultHttpClient` inclui `HttpClientBuilder` classes que podem ser usadas no lugar dos equivalentes do `HttpComponents` Apache para instrumentar chamadas HTTP de saída.

- `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`

Essas bibliotecas estão no submódulo [aws-xray-recorder-sdk-apache-http](#).

Você pode substituir as instruções de importação existentes pelo X-Ray equivalente para instrumentar todos os clientes ou usar o nome totalmente qualificado quando inicializar um cliente para instrumentar clientes específicos.

Example HttpClientBuilder

```
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
```

```
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.util.EntityUtils;
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
public String randomName() throws IOException {
    CloseableHttpClient httpClient = HttpClientBuilder.create().build();
    HttpGet httpGet = new HttpGet("http://names.example.com/api/");
    CloseableHttpResponse response = httpClient.execute(httpGet);
    try {
        HttpEntity entity = response.getEntity();
        InputStream inputStream = entity.getContent();
        ObjectMapper mapper = new ObjectMapper();
        Map<String, String> jsonMap = mapper.readValue(inputStream, Map.class);
        String name = jsonMap.get("name");
        EntityUtils.consume(entity);
        return name;
    } finally {
        response.close();
    }
}
```

Quando você instrumenta uma chamada para uma API subsequente da web, o X-Ray SDK para Java registra um subsegmento com informações sobre a solicitação e a resposta HTTP. O X-Ray usa o subsegmento para gerar um segmento inferido para a API remota.

Example Subsegmento para uma chamada HTTP downstream

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



```
}  
}
```

Example Segmento inferido para uma chamada HTTP de downstream

```
{  
  "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  
}
```

Rastrear consultas SQL com o X-Ray SDK para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Interceptores SQL

Instrumente consultas SQL adicionando o interceptor JDBC do X-Ray SDK para Java à configuração da fonte de dados.

- PostgreSQL: `com.amazonaws.xray.sql.postgres.TracingInterceptor`
- MySQL: `com.amazonaws.xray.sql.mysql.TracingInterceptor`

Esses interceptores estão nos [aws-xray-recorder-sql-postgres](#) e [aws-xray-recorder-sql-mysql](#) submódulos, respectivamente. Eles implementam `org.apache.tomcat.jdbc.pool.JdbcInterceptor` e são compatíveis com grupos de conexão do Tomcat.

Note

Interceptores SQL não registram a consulta SQL em si dentro de subsegmentos para fins de segurança.

Para Spring, adicione o interceptor em um arquivo de propriedades e crie a fonte de dados com o `DataSourceBuilder` do Spring Boot.

Example **`src/main/java/resources/application.properties`** – interceptor PostgreSQL JDBC

```
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`** – fonte de dados

```
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.jdbc.DataSourceBuilder;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
```

```

import javax.servlet.Filter;
import javax.sql.DataSource;
import java.net.URL;

@Configuration
@EnableAutoConfiguration
@EnableJpaRepositories("myapp")
public class RdsWebConfig {

    @Bean
    @ConfigurationProperties(prefix = "spring.datasource")
    public DataSource dataSource() {
        logger.info("Initializing PostgreSQL datasource");
        return DataSourceBuilder.create()
            .driverClassName("org.postgresql.Driver")
            .url("jdbc:postgresql://" + System.getenv("RDS_HOSTNAME") + ":" +
System.getenv("RDS_PORT") + "/ebdb")
            .username(System.getenv("RDS_USERNAME"))
            .password(System.getenv("RDS_PASSWORD"))
            .build();
    }
    ...
}

```

Para Tomcat, chame `setJdbcInterceptors` na fonte de dados JDBC com uma referência à classe do X-Ray SDK para Java.

Exemplo **src/main/myapp/model.java** – fonte de dados

```

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");

```

A biblioteca da fonte de dados JDBC Tomcat está incluída no X-Ray SDK para Java, mas é possível declará-la como uma dependência fornecida para documentar que você a usa.

Example **pom.xml** – fonte de dados JDBC

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jdbc</artifactId>
  <version>8.0.36</version>
  <scope>provided</scope>
</dependency>
```

Decorador de rastreamento SQL nativo

- Adicione [aws-xray-recorder-sdk-sql](#) às suas dependências.
- Decore sua fonte de dados, conexão ou declaração Decorate banco de dados.

```
dataSource = TracingDataSource.decorate(dataSource)
connection = TracingConnection.decorate(connection)
statement = TracingStatement.decorateStatement(statement)
preparedStatement = TracingStatement.decoratePreparedStatement(preparedStatement,
    sql)
callableStatement = TracingStatement.decorateCallableStatement(callableStatement,
    sql)
```

Gerar subsegmentos personalizados com o X-Ray SDK para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Os subsegmentos estendem o [segmento](#) de um rastreamento com detalhes sobre o trabalho realizado para atender a uma solicitação. Sempre que você faz uma chamada com um cliente instrumentado, o X-Ray SDK registra as informações geradas em um subsegmento. Você pode criar

subsegmentos adicionais para agrupar outros subsegmentos, medir o desempenho de uma seção do código ou registrar anotações e metadados.

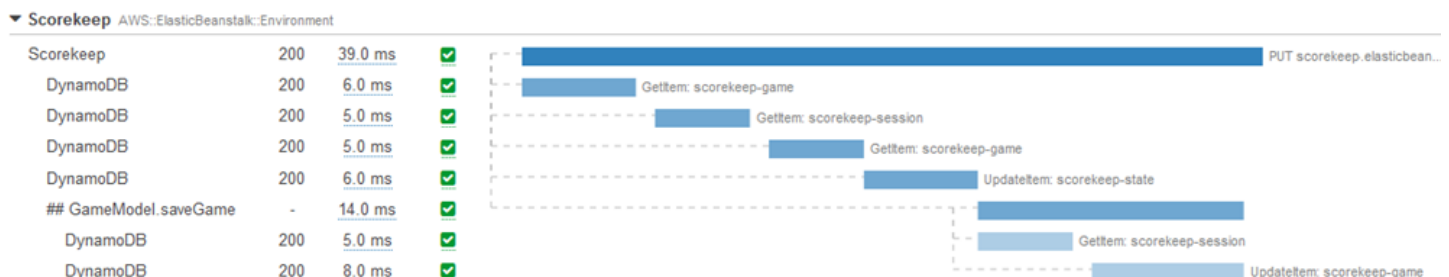
Para gerenciar subsegmentos, use os métodos `beginSubsegment` e `endSubsegment`.

Example `GameModel.java` - subsegmento personalizado

```
import com.amazonaws.xray.AWSXRay;

...
public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("Save Game");
    try {
        // check session
        String sessionId = game.getSession();
        if (sessionModel.loadSession(sessionId) == null ) {
            throw new SessionNotFoundException(sessionId);
        }
        mapper.save(game);
    } catch (Exception e) {
        subsegment.addException(e);
        throw e;
    } finally {
        AWSXRay.endSubsegment();
    }
}
```

Neste exemplo, o código dentro do subsegmento carrega a sessão do jogo do DynamoDB com um método no modelo de sessão e usa o mapeador do DynamoDB para AWS SDK para Java salvar o jogo. O encapsulamento desse código em um subsegmento transforma as chamadas do DynamoDB em filhas do subsegmento `Save Game` na visualização de rastreamento no console.



Se o código no seu subsegmento lança exceções verificadas, encapsule-o em um bloco `try` e chame `AWSXRay.endSubsegment()` em um bloco `finally` para garantir que o subsegmento

esteja sempre fechado. Se um subsegmento não estiver fechado, o segmento pai não poderá ser concluído e não será enviado para o X-Ray.

Para código que não lança exceções verificadas, você pode transmitir o código para `AWSXRay.CreateSubsegment` como uma função do Lambda.

Example Função de subsegmento do Lambda

```
import com.amazonaws.xray.AWSXRay;  
  
AWSXRay.createSubsegment("getMovies", (subsegment) -> {  
    // function code  
});
```

Quando você cria um subsegmento dentro de um segmento ou outro subsegmento, o X-Ray SDK para Java gera um ID para ele e registra a hora de início e de término.

Example Subsegmento com metadados

```
"subsegments": [{  
  "id": "6f1605cd8a07cb70",  
  "start_time": 1.480305974194E9,  
  "end_time": 1.4803059742E9,  
  "name": "Custom subsegment for UserModel.saveUser function",  
  "metadata": {  
    "debug": {  
      "test": "Metadata string from UserModel.saveUser"  
    }  
  },  
},
```

Para programação assíncrona e com vários threads, você deve passar manualmente o subsegmento para o método `endSubsegment()` para garantir que ele seja fechado corretamente, pois o contexto do X-Ray pode ser modificado durante a execução assíncrona. Se um subsegmento assíncrono for fechado após o fechamento do segmento pai, esse método transmitirá automaticamente o segmento inteiro para o daemon do X-Ray.

Example Subsegmento assíncrono

```
@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();
}
```

Adicionar anotações e metadados aos segmentos com o X-Ray SDK para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode registrar informações adicionais sobre solicitações, o ambiente ou o aplicativo com anotações e metadados. Também é possível adicionar anotações e metadados aos segmentos que o X-Ray SDK cria ou aos subsegmentos personalizados que você cria.

Anotações são pares de chave-valor com valores booleanos, de string ou número. As anotações são indexadas para serem usadas com [expressões de filtro](#). Use anotações para registrar dados que você deseja usar para agrupar rastreamentos no console ou ao chamar a API [GetTraceSummaries](#).

Metadados são pares chave-valor que podem ter valores de qualquer tipo, incluindo objetos e listas, mas não são indexados para uso com expressões de filtro. Use metadados para registrar dados adicionais que você deseja armazenar no rastreamento e não precisa usar com a pesquisa.

Além de anotações e metadados, você também pode [registrar strings de ID de usuário](#) em segmentos. Os usuários são registrados em um campo separado nos segmentos e são indexados para uso com a pesquisa.

Seções

- [Registrar anotações com o X-Ray SDK para Java](#)
- [Registrar metadados com o X-Ray SDK para Java](#)
- [Gravando usuário IDs com o X-Ray SDK for Java](#)

Registrar anotações com o X-Ray SDK para Java

Use anotações para registrar informações em segmentos ou subsegmentos que você deseja indexar para pesquisa.

Requisitos de anotação

- Chaves: a chave para uma anotação do X-Ray pode ter até 500 caracteres alfanuméricos. Você não pode usar espaços ou símbolos que não sejam um ponto (.)
- Valores: o valor de uma anotação do X-Ray pode ter até 1.000 caracteres Unicode.
- O número de anotações: você pode usar até cinquenta anotações por rastreamento.

Como registrar anotações

1. Obtenha uma referência para o segmento ou subsegmento atual no AWSXRay.

```
import com.amazonaws.xray.AWSXRay;  
import com.amazonaws.xray.entities.Segment;  
...  
Segment document = AWSXRay.getCurrentSegment();
```

ou

```
import com.amazonaws.xray.AWSXRay;  
import com.amazonaws.xray.entities.Subsegment;  
...  
Subsegment document = AWSXRay.getCurrentSubsegment();
```

2. Chame `putAnnotation` com uma chave de string e um valor de número, string ou booleano.


```
document.putAnnotation("mykey", "my value");
```

O exemplo a seguir mostra como chamar `putAnnotation` com uma chave `String` que inclui um ponto e um valor booleano, numérico ou de string.

```
document.putAnnotation("testkey.test", "my value");
```

O SDK registra anotações como pares de chave-valor em um objeto `annotations` no documento de segmentos. Chamar `putAnnotation` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Para encontrar rastreamentos que têm anotações com valores específicos, use a palavra-chave `annotation[key]` em uma [expressão de filtro](#).

Exemplo [src/main/java/scorekeep/GameModel.java](#): anotações e metadados

```
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();
    }
}
```

```
}
```

Registrar metadados com o X-Ray SDK para Java

Use metadados para registrar informações em segmentos ou subsegmentos dos quais você não precisa indexados para pesquisa. Valores de metadados podem ser strings, números, booleanos ou qualquer objeto que possa ser serializado em uma matriz ou objeto JSON.

Como registrar metadados

1. Obtenha uma referência para o segmento ou subsegmento atual no AWSXRay.

```
import com.amazonaws.xray.AWSXRay;  
import com.amazonaws.xray.entities.Segment;  
...  
Segment document = AWSXRay.getCurrentSegment();
```

ou

```
import com.amazonaws.xray.AWSXRay;  
import com.amazonaws.xray.entities.Subsegment;  
...  
Subsegment document = AWSXRay.getCurrentSubsegment();
```

2. Chame `putMetadata` com um namespace de string e uma chave de string e um valor booleano, de número, string ou objeto.

```
document.putMetadata("my namespace", "my key", "my value");
```

or

Chame `putMetadata` com apenas uma chave e valor.

```
document.putMetadata("my key", "my value");
```

Se você não especificar um namespace, o SDK usará `default`. Chamar `putMetadata` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Exemplo [src/main/java/scorekeep/GameModel.java](#): anotações e metadados

```
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();
    }
}
```

Gravando usuário IDs com o X-Ray SDK for Java

Registre o usuário IDs nos segmentos da solicitação para identificar o usuário que enviou a solicitação.

Para gravar usuário IDs

1. Obtenha uma referência para o segmento atual em AWSXRay.

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
...
Segment document = AWSXRay.getCurrentSegment();
```

2. Chame setUser com um ID de string do usuário que enviou a solicitação.

```
document.setUser("U12345");
```

Você pode acionar `setUser` em seus controladores para registrar o ID de usuário assim que o aplicativo começar a processar uma solicitação. Se pretende usar o segmento apenas para definir o ID de usuário, você pode vincular as chamadas em uma única linha.

Example [src/main/java/scorekeep/MoveController.java](#) — ID do usuário

```
import com.amazonaws.xray.AWSXRay;
...
@RequestMapping(value="/{userId}", method=RequestMethod.POST)
public Move newMove(@PathVariable String sessionId, @PathVariable String
gameId, @PathVariable String userId, @RequestBody String move) throws
SessionNotFoundException, GameNotFoundException, StateNotFoundException,
RulesException {
    AWSXRay.getCurrentSegment().setUser(userId);
    return moveFactory.newMove(sessionId, gameId, userId, move);
}
```

Para encontrar rastreamentos para um ID de usuário, use a `user` palavra-chave em uma [expressão de filtragem](#).

AWS X-Ray métricas para o X-Ray SDK for Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Este tópico descreve o AWS X-Ray namespace, as métricas e as dimensões. Você pode usar o X-Ray SDK for Java para publicar métricas sem amostragem da CloudWatch Amazon a partir

dos segmentos coletados do X-Ray. Essas métricas são derivadas da hora de início e término do segmento e dos sinalizadores de status de erro, falha e limitação. Use essas métricas de rastreamento para expor novas tentativas e problemas de dependência dentro de subsegmentos.

CloudWatch é um repositório de métricas. Uma métrica é o conceito fundamental CloudWatch e representa um conjunto de pontos de dados ordenado pelo tempo. Você (ou Serviços da AWS) publica pontos de dados de métricas CloudWatch e recupera estatísticas sobre esses pontos de dados como um conjunto ordenado de dados de séries temporais.

As métricas são definidas exclusivamente por um nome, um namespace e uma ou mais dimensões. Cada ponto de dados tem um timestamp e, opcionalmente, uma unidade de medida. Quando você solicita estatísticas, o fluxo de dados apresentado é identificado pelo namespace, pelo nome da métrica e pela dimensão.

Para obter mais informações sobre CloudWatch, consulte o [Guia CloudWatch do usuário da Amazon](#).

CloudWatch Métricas de X-Ray

O namespace `ServiceMetrics/SDK` inclui as métricas a seguir.

Métrica	Estatísticas disponíveis	Description	Unidades
Latency	Média, Mínimo Máximo, Contagem	A diferença entre a hora de início e de término. A média, o mínimo e o máximo descrevem a latência operacional. A contagem descreve a contagem de chamadas.	Milissegundos
ErrorRate	Média, Soma	A taxa de solicitações que falharam com um código de status 4xx Client Error,	Percentual

Métrica	Estatísticas disponíveis	Description	Unidades
		resultando em um erro.	
<code>FaultRate</code>	Média, Soma	A taxa de rastreamentos que falharam com um código de status 5xx <code>ServerError</code> , resultando em uma falha.	Percentual
<code>ThrottleRate</code>	Média, Soma	A taxa de rastreamentos limitados que retornam um código de status 429. Este é um subconjunto da métrica <code>ErrorRate</code> .	Percentual
<code>OkRate</code>	Média, Soma	A taxa de solicitações rastreadas que resultam em um código de status OK.	Percentual

CloudWatch Dimensões do X-Ray

Use as dimensões na tabela a seguir para refinar as métricas retornadas para aplicações Java instrumentadas do X-Ray.

Dimensão	Description
<code>ServiceType</code>	O tipo do serviço, por exemplo, <code>AWS::EC2::Instance</code> ou <code>NONE</code> , se não for conhecido.
<code>ServiceName</code>	O nome canônico do serviço.

Ativar CloudWatch métricas de X-Ray

Use o procedimento a seguir para habilitar métricas de rastreamento na aplicação Java instrumentada.

Para configurar métricas de rastreamento

1. Adicione o pacote `aws-xray-recorder-sdk-metrics` como uma dependência do Apache Maven. Para obter mais informações, consulte [Submódulos do X-Ray SDK para Java](#).
2. Ative um novo `MetricsSegmentListener()` como parte da compilação global de gravador.

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. Implante o CloudWatch agente para coletar métricas usando o Amazon Elastic Compute Cloud (Amazon EC2), o Amazon Elastic Container Service (Amazon ECS) ou o Amazon Elastic Kubernetes Service (Amazon EKS):

- Para configurar a Amazon EC2, consulte [Instalação do CloudWatch agente](#).
 - Para configurar o Amazon ECS, consulte [Monitorar contêineres do Amazon ECS usando o Container Insights](#).
 - Para configurar o Amazon EKS, consulte [Instalar o CloudWatch agente usando o complemento Amazon CloudWatch Observability EKS](#).
4. Configure o SDK para se comunicar com o CloudWatch agente. Por padrão, o SDK se comunica com o CloudWatch agente no endereço. 127.0.0.1 É possível configurar endereços alternativos definindo a variável de ambiente ou a propriedade do Java como `address:port`.

Example Variável de ambiente

```
AWS_XRAY_METRICS_DAEMON_ADDRESS=address:port
```

Example Propriedade do Java

```
com.amazonaws.xray.metrics.daemonAddress=address:port
```

Para validar a configuração

1. Faça login no Console de gerenciamento da AWS e abra o CloudWatch console em <https://console.aws.amazon.com/cloudwatch/>.
2. Abra a guia Métricas para observar o influxo de suas métricas.
3. (Opcional) No CloudWatch console, na guia Registros, abra o grupo de ServiceMetricsSDK registros. Procure um fluxo de log que corresponda às métricas do host e confirme as mensagens de log.

Passar o contexto do segmento entre threads em um aplicativo multithreaded

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais

informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando você cria um novo thread em seu aplicativo, o AWSXRayRecorder não mantém uma referência à [entidade](#) do segmento atual ou subsegmento. Se você usar um cliente instrumentado no novo thread, o SDK tentará gravar em um segmento que não existe, causando uma.

[SegmentNotFoundException](#)

Para evitar o lançamento de exceções durante o desenvolvimento, você pode configurar o gravador com um [ContextMissingStrategy](#) que solicita que ele registre um erro em vez disso. Você pode configurar a estratégia no código com [SetContextMissingStrategy](#), ou configurar opções equivalentes com uma [variável de ambiente](#) ou [propriedade do sistema](#).

Uma maneira de resolver o erro é usar um novo segmento chamando [beginSegment](#) quando iniciar o thread e [endSegment](#) quando fechá-lo. Isso funciona se você estiver instrumentando código que não seja executado em uma solicitação HTTP, como código que é executado quando seu aplicativo é iniciado.

Se usar vários threads para lidar com solicitações de entrada, você poderá passar o segmento ou subsegmento atual para o novo thread e fornecê-lo ao gravador global. Isso garante que as informações gravadas no novo thread sejam associadas ao mesmo segmento como o restante das informações gravadas sobre essa solicitação. Quando o segmento estiver disponível no novo thread, você poderá executar qualquer executável com acesso ao contexto desse segmento usando o método `segment.run(() -> { ... })`.

Consulte [Usar clientes instrumentais em threads de operador](#) para ver um exemplo.

Usar o X-Ray com programação assíncrona

O X-Ray SDK for Java pode ser usado em programas Java assíncronos com.

[SegmentContextExecutors](#) O `SegmentContextExecutor` implementa a interface do `Executor`, o que significa que ela pode ser passada para todas as operações assíncronas de um. [CompletableFuture](#) Isso garante que todas as operações assíncronas sejam executadas com o segmento correto no respectivo contexto.

Example Exemplo de App.java: Passando SegmentContextExecutor para 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 com o Spring e X-Ray SDK para Java

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Este tópico descreve como usar o X-Ray SDK e o Spring Framework para instrumentar a aplicação sem alterar a lógica central. Isso significa que agora existe uma forma não invasiva de instrumentar seus aplicativos executados remotamente. AWS

Para habilitar a AOP no spring

1. [Configure o Spring](#)
2. [Adicionar um filtro de rastreamento ao aplicativo](#)
3. [Anotar o seu código ou implemente uma interface](#)
4. [Ative o X-Ray em seu aplicativo](#)

Configurar o Spring

Você pode usar o Maven ou o Gradle para configurar o Spring para usar AOP para instrumentar seu aplicativo.

Se você usar o Maven para compilar seu aplicativo, adicione a dependência a seguir no arquivo `pom.xml`.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-xray-recorder-sdk-spring</artifactId>
  <version>2.11.0</version>
</dependency>
```

Para o Gradle, adicione a seguinte dependência no arquivo `build.gradle`.

```
compile 'com.amazonaws:aws-xray-recorder-sdk-spring:2.11.0'
```

Configurar o Spring Boot

Além da dependência do Spring descrita na seção anterior, se você estiver usando o Spring Boot, adicione a seguinte dependência se ela ainda não estiver no classpath.

Maven:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-aop</artifactId>
  <version>2.5.2</version>
</dependency>
```

Gradle:

```
compile 'org.springframework.boot:spring-boot-starter-aop:2.5.2'
```

Adicionar um filtro de rastreamento ao aplicativo

Adicione um `Filter` à classe `WebConfig`. Passe o nome do segmento para o construtor [AWSXRayServletFilter](#) como uma string. Para obter mais informações sobre filtros de rastreamento e instrumentar solicitações de entrada, consulte [Rastrear solicitações recebidas com o X-Ray SDK para Java](#).

Example src/main/java/myapp/WebConfig.java - primavera

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;

@Configuration
public class WebConfig {

    @Bean
    public Filter TracingFilter() {
        return new AWSXRayServletFilter("Scorekeep");
    }
}
```

Suporta a Jakarta

O Spring 6 usa [Jakarta](#) em vez de Javax na Enterprise Edition. Para oferecer compatibilidade com esse novo namespace, o X-Ray criou um conjunto paralelo de classes que residem em seu próprio namespace Jakarta.

Para as classes de filtro, substitua javax por jakarta. Ao configurar uma estratégia de nomeação de segmentos, adicione jakarta antes do nome da classe da estratégia, como no seguinte exemplo:

```
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"));
    }
}
```

Anotar o código ou implementar uma interface

Anote suas classes com a anotação `@XRayEnabled` ou implemente a interface `XRayTraced`. Isso informa ao sistema de AOP para encapsular as funções da classe afetada para a instrumentação do X-Ray.

Ativar o X-Ray na aplicação

Para ativar o rastreamento do X-Ray na aplicação, o código deve estender a classe abstrata `BaseAbstractXRayInterceptor` substituindo os métodos a seguir.

- `generateMetadata`: esta função permite a personalização dos metadados anexados ao rastreamento da função atual. Por padrão, o nome de classe da função de execução é registrada nos metadados. Você pode adicionar mais dados se precisar de informações adicionais.
- `xrayEnabledClasses`: esta função está vazia e deve permanecer assim. Ela serve como host para um `pointcut`, instruindo o interceptor sobre quais métodos devem ser encapsulados. Defina o `pointcut` especificando quais das classes que estão anotadas com `@XRayEnabled` serão rastreadas. A seguinte instrução de `pointcut` informa ao interceptor para encapsular todos os beans do controlador anotados com a anotação `@XRayEnabled`.

```
@Pointcut("@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)")
```

Se seu projeto estiver usando o Spring Data JPA, considere a possibilidade de estender de `AbstractXRayInterceptor` em vez de `BaseAbstractXRayInterceptor`.

Exemplo

O código a seguir estende a classe abstrata `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() {}

}
```

O código a seguir é uma classe que será instrumentada pelo X-Ray.

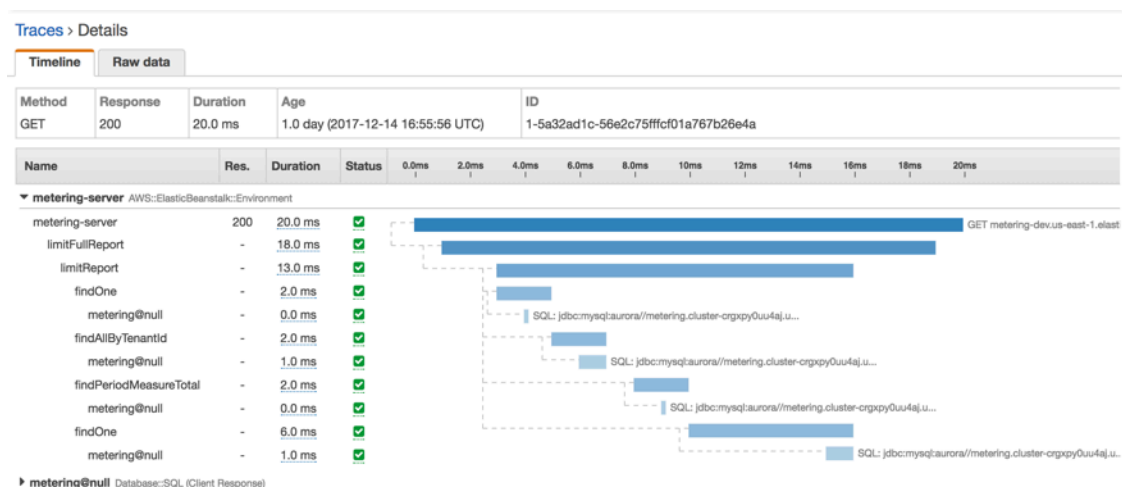
```
@Service
@XRayEnabled
public class MyServiceImpl implements MyService {
    private final MyEntityRepository myEntityRepository;

    @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());
        }
    }
}
```

Se você tiver configurado seu aplicativo corretamente, deverá ver a pilha de chamadas completa do aplicativo, do controlador até as chamadas de serviço, conforme mostrado na captura de tela do console a seguir.



Trabalho com Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Há duas maneiras de instrumentar um aplicativo Node.js para enviar rastreamentos ao X-Ray:

- [AWS Distro for OpenTelemetry JavaScript](#) — Uma AWS distribuição que fornece um conjunto de bibliotecas de código aberto para enviar métricas e rastreamentos correlacionados para várias soluções de AWS monitoramento, incluindo Amazon e Amazon OpenSearch Service CloudWatch AWS X-Ray, por meio do [AWS Distro](#) for Collector. OpenTelemetry
- [AWS X-Ray SDK para Node.js](#) — Um conjunto de bibliotecas para gerar e enviar rastreamentos para o X-Ray por meio do [daemon X-Ray](#).

Para obter mais informações, consulte [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#).

AWS Distro para OpenTelemetry JavaScript

Com o AWS Distro para OpenTelemetry (ADOT) JavaScript, você pode instrumentar as aplicações uma vez e enviar métricas e rastreamentos correlacionados para várias soluções de monitoramento da AWS como o Amazon CloudWatch, o AWS X-Ray e o Amazon OpenSearch Service. O uso do X-Ray com o AWS Distro para OpenTelemetry requer dois componentes: um SDK do OpenTelemetry habilitado para uso com o X-Ray e o coletor AWS Distro para OpenTelemetry habilitado para uso com o X-Ray.

Para começar, consulte a documentação do [AWS Distro para OpenTelemetry JavaScript](#).

Note

O ADOT JavaScript é compatível com todas as aplicações Node.js do lado do servidor. O ADOT JavaScript não consegue exportar dados para o X-Ray por meio de clientes de navegador.

Para obter mais informações sobre como usar o AWS Distro para OpenTelemetry com o AWS X-Ray e outros Serviços da AWS, consulte [AWS Distro para OpenTelemetry](#) ou a [documentação do AWS Distro para OpenTelemetry](#).

Para obter mais informações sobre compatibilidade de idiomas e uso, consulte [AWS Observability no GitHub](#).

AWS X-Ray SDK para Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O X-Ray SDK para Node.js é uma biblioteca para aplicações web Express e funções do Lambda em Node.js que fornece classes e métodos para gerar e enviar dados de rastreamento ao daemon do X-Ray. Os dados de rastreamento incluem informações sobre solicitações HTTP recebidas pelo aplicativo e chamadas que o aplicativo faz para serviços downstream usando o AWS SDK ou clientes HTTP.

 Note

O SDK do X-Ray para Node.js é um projeto de código aberto que é compatível com Node.js nas versões 14.x e superiores. Você pode acompanhar o projeto e enviar problemas e pull requests em GitHub: github.com/aws/aws-xray-sdk-node

Se você usa o Express, comece [adicionando o SDK como middleware](#) em seu servidor de aplicativo para rastrear solicitações recebidas. O middleware cria um [segmento](#) para cada solicitação rastreada e conclui o segmento quando a resposta é enviada. Embora o segmento esteja aberto, você pode usar os métodos do cliente do SDK para adicionar informações ao segmento e criar subsegmentos para rastrear as chamadas subsequentes. O SDK também registra automaticamente exceções que seu aplicativo lança enquanto o segmento está aberto.

Para funções do Lambda chamadas por um aplicativo ou um serviço instrumentado, o Lambda lê o [cabeçalho de rastreamento](#) e rastreia automaticamente as solicitações amostradas. Para outras funções, você pode [configurar o Lambda](#) para amostrar e rastrear solicitações recebidas. Em ambos os casos, o Lambda cria o segmento e o fornece ao X-Ray SDK.

 Note

No Lambda, o X-Ray SDK é opcional. Se você não o usar em sua função, mesmo assim o mapa de serviço incluirá um nó para o serviço Lambda e um para cada função do Lambda. Ao adicionar o SDK, você pode instrumentar o código da função para adicionar subsegmentos ao segmento de função registrado pelo Lambda. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Em seguida, use o X-Ray SDK para Node.js para [instrumentar seu AWS SDK JavaScript em clientes Node.js](#). Sempre que você faz uma chamada para um downstream AWS service (Serviço da AWS) ou recurso com um cliente instrumentado, o SDK registra as informações sobre a chamada em um subsegmento. Serviços da AWS e os recursos que você acessa nos serviços aparecem como nós downstream no mapa de rastreamento para ajudá-lo a identificar erros e problemas de limitação em conexões individuais.

O X-Ray SDK para Node.js também fornece instrumentação para chamadas downstream para consultas HTTP Web APIs e SQL. [Encapsule seu cliente HTTP no método de captura do SDK](#) para

registrar informações sobre chamadas HTTP de saída. Para clientes SQL, [use o método de captura para o seu tipo de banco de dados](#).

O middleware aplica regras de amostragem a solicitações recebidas para determinar quais solicitações devem ser rastreadas. Você pode [configurar o X-Ray SDK para Node.js para](#) ajustar o comportamento de amostragem ou registrar informações sobre os recursos AWS computacionais nos quais seu aplicativo é executado.

Registre informações adicionais sobre as solicitações e o trabalho que o aplicativo faz em [anotações e metadados](#). Anotações são simples pares de chave-valor que são indexados para serem usados com [expressões de filtro](#) para que você possa pesquisar rastreamentos que contêm dados específicos. As entradas de metadados são menos restritivas e podem registrar matrizes e objetos inteiros: tudo o que pode ser serializado em JSON.

Anotações e metadados

Anotações e metadados são textos arbitrários que você adiciona aos segmentos com o X-Ray SDK. As anotações são indexadas para serem usadas com expressões de filtro. Os metadados não são indexados, mas podem ser visualizados no segmento bruto com o console ou a API do X-Ray. Qualquer pessoa à qual você conceder acesso de leitura ao X-Ray poderá visualizar esses dados.

Quando há uma grande quantidade de clientes instrumentados no código, um único segmento de solicitação pode conter um grande número de subsegmentos, um para cada chamada feita com um cliente instrumentado. Você pode organizar e agrupar subsegmentos integrando chamadas de clientes em [subsegmentos personalizados](#). Você pode criar um subsegmento personalizado para uma função inteira ou qualquer seção de código e registrar metadados e anotações no subsegmento em vez de gravar tudo no segmento principal.

Para documentação de referência sobre as classes e os métodos do SDK, consulte a [Referência de API do AWS X-Ray SDK para Node.js](#).

Requisitos

O X-Ray SDK para Node.js requer o Node.js e as seguintes bibliotecas:

- `atomic-batcher`: 1.0.2
- `cls-hooked`: 4.2.2

- pkginfo: 0.4.0
- semver: 5.3.0

O SDK obtém essas bibliotecas quando você o instala com NPM.

Para rastrear clientes AWS SDK, o X-Ray SDK para Node.js exige uma versão mínima do AWS SDK para JavaScript Node.js.

- aws-sdk: 2.7.15

Gerenciamento de dependências

O X-Ray SDK para Node.js está disponível no NPM.

- Pacote: [aws-xray-sdk](#)

Para desenvolvimento local, instale o SDK no diretório de projetos com npm.

```
~/nodejs-xray$ npm install aws-xray-sdk
aws-xray-sdk@3.3.3
  ### aws-xray-sdk-core@3.3.3
  # ### @aws-sdk/service-error-classification@3.15.0
  # ### @aws-sdk/types@3.15.0
  # ### @types/cls-hooked@4.3.3
  # # ### @types/node@15.3.0
  # ### atomic-batcher@1.0.2
  # ### cls-hooked@4.2.2
  # # ### async-hook-jl@1.7.6
  # # # ### stack-chain@1.3.7
  # # ### emitter-listener@1.1.2
  # #   ### shimmer@1.2.1
  # ### semver@5.7.1
  ### aws-xray-sdk-express@3.3.3
  ### aws-xray-sdk-mysql@3.3.3
  ### aws-xray-sdk-postgres@3.3.3
```

Use a opção `--save` para salvar o SDK como uma dependência no `package.json` no seu aplicativo.

```
~/nodejs-xray$ npm install aws-xray-sdk --save
```

```
aws-xray-sdk@3.3.3
```

Se a aplicação tiver alguma dependência cujas versões entrem em conflito com as dependências do X-Ray SDK, ambas as versões serão instaladas para garantir a compatibilidade. Para obter mais detalhes, consulte a [documentação oficial do NPM para resolução de dependências](#).

Amostras de Node.js

Trabalhe com o AWS X-Ray SDK para Node.js para ter uma end-to-end visão das solicitações à medida que elas percorrem seus aplicativos Node.js.

- [Aplicação de amostra Node.js](#) ativada GitHub.

Configurar o X-Ray SDK para Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode configurar o X-Ray SDK para Node.js com plug-ins para incluir informações sobre o serviço em que a aplicação é executada, modificar o comportamento de amostragem padrão ou adicionar regras de amostragem que se aplicam a solicitações para caminhos específicos.

Seções

- [Plug-ins de serviço](#)
- [Regras de amostragem](#)
- [Registro em log](#)
- [Endereço do daemon do X-Ray](#)
- [Variáveis de ambiente](#)

Plug-ins de serviço

Use plugins para registrar informações sobre o serviço que hospeda o aplicativo.

Plugins

- Amazon EC2 — EC2Plugin adiciona o ID da instância, a zona de disponibilidade e o grupo de CloudWatch registros.
- Elastic Beanstalk: o ElasticBeanstalkPlugin adiciona o nome do ambiente, o rótulo da versão e o ID de implantação.
- Amazon ECS: o ECSPPlugin adiciona o ID do contêiner.

Para usar um plug-in, configure o cliente do X-Ray SDK para Node.js usando o método `config`.

Example app.js – plug-ins

```
var AWSXRay = require('aws-xray-sdk');  
AWSXRay.config([AWSXRay.plugins.EC2Plugin, AWSXRay.plugins.ElasticBeanstalkPlugin]);
```

O SDK também usa as configurações do plug-in para definir o campo `origin` no segmento. Isso indica o tipo de AWS recurso que executa seu aplicativo. Quando você usa vários plug-ins, o SDK usa a seguinte ordem de resolução para determinar a origem: ElasticBeanstalk > EKS > ECS > EC2.

Regras de amostragem

O SDK usa as regras de amostragem que você define no console do X-Ray para determinar quais solicitações serão registradas. A regra padrão rastreia a primeira solicitação a cada segundo e 5% de todas as solicitações adicionais em todos os serviços que enviam rastreamentos ao X-Ray. [Crie regras adicionais no console do X-Ray](#) para personalizar a quantidade de dados registrados para cada uma dos aplicativos.

O SDK aplica regras personalizadas na ordem em que elas estão definidas. Se uma solicitação corresponder a várias regras personalizadas, o SDK aplicará somente a primeira regra.

Note

Se o SDK não conseguir acessar o X-Ray para obter regras de amostragem, ele reverterá para uma regra local padrão da primeira solicitação a cada segundo e 5% de todas as solicitações adicionais por host. Isso pode ocorrer se o host não tiver permissão para chamar

a amostragem APIs ou não conseguir se conectar ao daemon X-Ray, que atua como um proxy TCP para chamadas de API feitas pelo SDK.

Você também pode configurar o SDK para carregar regras de amostragem de um documento JSON. O SDK pode usar regras locais como backup para casos em que a amostragem do X-Ray não está disponível ou usar exclusivamente regras locais.

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
  }
}
```

Este exemplo define uma regra personalizada e uma regra padrão. A regra personalizada aplica uma taxa de amostragem de 5% sem um número mínimo de solicitações para rastrear os caminhos em `/api/move/`. A regra padrão rastreia a primeira solicitação a cada segundo e 10% das solicitações adicionais.

A desvantagem de definir regras localmente é que o destino fixo é aplicado por instância do gravador de forma independente, em vez de ser gerenciado pelo serviço X-Ray. À medida que você implanta mais hosts, a taxa fixa é multiplicada, dificultando o controle da quantidade de dados registrados.

Ativado AWS Lambda, você não pode modificar a taxa de amostragem. Se sua função for chamada por um serviço instrumentado, as chamadas que geraram solicitações que foram amostradas por esse serviço serão registradas pelo Lambda. Se o rastreamento ativo estiver habilitado e nenhum cabeçalho de rastreamento estiver presente, o Lambda tomará a decisão de amostragem.

Para configurar regras de backup, instrua o X-Ray SDK para Node.js a carregar regras de amostragem de um arquivo com `setSamplingRules`.

Example app.js – regras de amostragem de um arquivo

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
```

Você também pode definir suas regras em código e transmiti-los a `setSamplingRules` como um objeto.

Example app.js – regras de amostra de um arquivo

```
var AWSXRay = require('aws-xray-sdk');
var rules = {
  "rules": [ { "description": "Player moves.", "service_name": "*", "http_method": "*",
"url_path": "/api/move/*", "fixed_target": 0, "rate": 0.05 } ],
  "default": { "fixed_target": 1, "rate": 0.1 },
  "version": 1
}

AWSXRay.middleware.setSamplingRules(rules);
```

Para usar apenas regras locais, chame `disableCentralizedSampling`.

```
AWSXRay.middleware.disableCentralizedSampling()
```

Registro em log

Para registrar a saída do SDK, chame `AWSXRay.setLogger(logger)`, em que `logger` é um objeto que fornece métodos de registro padrão (`warn`, `info` etc.).

Por padrão, o SDK registrará as mensagens de erro em log no console usando os métodos padrão no objeto do console. O nível de log do registrador integrado pode ser definido usando as variáveis de ambiente `AWS_XRAY_DEBUG_MODE` ou `AWS_XRAY_LOG_LEVEL`. Para obter uma lista de valores de nível de log válidos, consulte [Variáveis de ambiente](#).

Se quiser fornecer um formato ou destino diferente para os logs, você pode fornecer ao SDK sua própria implementação da interface do registrador, conforme mostrado abaixo. Qualquer objeto que

implemente essa interface pode ser usado. Isso significa que muitas bibliotecas de registro em log (por exemplo, Winston) podem ser usadas e passadas diretamente para o SDK.

Example app.js – registro

```
var AWSXRay = require('aws-xray-sdk');

// Create your own logger, or instantiate one using a library.
var logger = {
  error: (message, meta) => { /* logging code */ },
  warn: (message, meta) => { /* logging code */ },
  info: (message, meta) => { /* logging code */ },
  debug: (message, meta) => { /* logging code */ }
}

AWSXRay.setLogger(logger);
AWSXRay.config([AWSXRay.plugins.EC2Plugin]);
```

Chame `setLogger` antes de executar outros métodos de configuração para garantir que você capture a saída dessas operações.

Endereço do daemon do X-Ray

Se o daemon do X-Ray escuta em uma porta ou host diferente de `127.0.0.1:2000`, você pode configurar o X-Ray SDK para Node.js para enviar dados de rastreamento a um endereço diferente.

```
AWSXRay.setDaemonAddress('host:port');
```

Você pode especificar o host por nome ou IPv4 endereço.

Example app.js – endereço daemon

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('daemonhost:8082');
```

Se você tiver configurado o daemon para escutar em portas diferentes para TCP e UDP, poderá especificar ambas na configuração de endereço do daemon.

Example app.js – endereço daemon em portas separadas

```
var AWSXRay = require('aws-xray-sdk');
```



```
AWSXRay.setDaemonAddress('tcp:daemonhost:8082 udp:daemonhost:8083');
```

Você também pode definir o endereço daemon usando a `AWS_XRAY_DAEMON_ADDRESS` [variável de ambiente](#).

Variáveis de ambiente

Você pode usar variáveis de ambiente para configurar o X-Ray SDK para Node.js. O SDK é compatível com as variáveis a seguir.

- `AWS_XRAY_CONTEXT_MISSING`: defina como `RUNTIME_ERROR` para lançar exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.

Valores válidos

- `RUNTIME_ERROR`: lance uma exceção de tempo de execução.
- `LOG_ERROR`: registre um erro e continue (padrão).
- `IGNORE_ERROR`: ignore o erro e continue.

Erros relativos a segmentos ou subsegmentos ausentes poderão ocorrer quando você tentar usar um cliente instrumentado no código de inicialização que é executado quando nenhuma solicitação estiver aberta ou em um código que gere um novo thread.

- `AWS_XRAY_DAEMON_ADDRESS`: defina o host e a porta do receptor do daemon do X-Ray. Por padrão, o SDK usa `127.0.0.1:2000` para dados de rastreamento (UDP) e para amostragem (TCP). Use essa variável se você tiver configurado o daemon para [escutar em uma porta diferente](#) ou se ele estiver sendo executado em um host diferente.

Formato

- Mesma porta: *address:port*
- Portas diferentes: `tcp:address:port udp:address:port`
- `AWS_XRAY_DEBUG_MODE`: defina como `TRUE` para configurar o SDK para enviar logs ao console, em nível de debug.
- `AWS_XRAY_LOG_LEVEL` : defina um nível de log para o registrador padrão. Os valores válidos são `debug`, `info`, `warn`, `error` e `silent`. Esse valor é ignorado quando `AWS_XRAY_DEBUG_MODE` definido como `TRUE`.
- `AWS_XRAY_TRACING_NAME`: defina um nome de serviço para o SDK usar para segmentos. Sobreponha o nome do segmento que você [definiu no middleware Express](#).

Rastrear solicitações de entrada com o X-Ray SDK para Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Você pode usar o X-Ray SDK para Node.js para rastrear solicitações HTTP recebidas que seus aplicativos Express e Restify atendem EC2 em uma instância na Amazon AWS Elastic Beanstalk ou no Amazon EC2 ECS.

O X-Ray SDK para Node.js fornece middleware para aplicações que usam os frameworks Express e Restify. Quando você adiciona o middleware do X-Ray ao aplicativo, o X-Ray SDK para Node.js cria um segmento para cada solicitação amostrada. Este segmento inclui tempo, método e disposição da solicitação HTTP. Instrumentação adicional cria subsegmentos sobre esse segmento.

Note

Para AWS Lambda funções, o Lambda cria um segmento para cada solicitação de amostra. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Cada segmento tem um nome que identifica o aplicativo no mapa de serviços. O segmento pode ser nomeado estaticamente ou você pode configurar o SDK para nomeá-lo dinamicamente com base no cabeçalho do host na solicitação de entrada. A nomeação dinâmica permite agrupar rastreamentos com base no nome de domínio na solicitação e aplicar um nome padrão se o nome não corresponder ao padrão esperado (por exemplo, se o cabeçalho do host for falsificado).

Solicitações encaminhadas

Se um balanceador de carga ou outro intermediário encaminhar uma solicitação para o aplicativo, o X-Ray obterá o IP do cliente do cabeçalho `X-Forwarded-For` na solicitação

em vez do IP de origem no pacote IP. O IP do cliente registrado para uma solicitação encaminhada pode ser falsificado; portanto, não é digno de confiança.

Quando uma solicitação é encaminhada, o SDK define um campo adicional no segmento para indicar isso. Se o segmento tiver o campo `x_forwarded_for` definido como `true`, isso significa que o IP do cliente foi obtido no cabeçalho `X-Forwarded-For` na solicitação HTTP.

O manipulador de mensagens cria um segmento para cada solicitação recebida com um bloco `http` que contém as seguintes informações:

- Método HTTP: GET, POST, PUT, DELETE etc.
- Endereço do cliente: o endereço IP do cliente que enviou a solicitação.
- Código de resposta: o código de resposta HTTP da solicitação concluída.
- Horário: a hora de início (quando a solicitação foi recebida) e a hora de término (quando a resposta foi enviada).
- Agente do usuário: o `user-agent` da solicitação.
- Tamanho do conteúdo: o `content-length` da resposta.

Seções

- [Rastreamento de solicitações recebidas com Express](#)
- [Rastreamento de solicitações recebidas com Restify](#)
- [Configurar uma estratégia de nomeação de segmentos](#)

Rastreamento de solicitações recebidas com Express

Para usar o middleware Express, inicie o cliente do SDK e use o middleware retornado pela função `express.openSegment` antes de definir as rotas.

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());
```

Depois de definir as rotas, use a saída de `express.closeSegment`, como mostrado, para lidar com quaisquer erros retornados pelo X-Ray SDK para Node.js.

Rastreamento de solicitações recebidas com Restify

Para usar o middleware Restify, inicie o cliente do SDK e execute `enable`. Transmita seu servidor Restify e o nome do segmento.

Example app.js – restify

```
var AWSXRay = require('aws-xray-sdk');  
var AWSXRayRestify = require('aws-xray-sdk-restify');  
  
var restify = require('restify');  
var server = restify.createServer();  
AWSXRayRestify.enable(server, 'MyApp');  
  
server.get('/', function (req, res) {  
  res.render('index');  
});
```

Configurar uma estratégia de nomeação de segmentos

AWS X-Ray usa um nome de serviço para identificar seu aplicativo e diferenciá-lo dos outros aplicativos, bancos de dados APIs, AWS recursos externos e que seu aplicativo usa. Quando o X-Ray SDK gera segmentos para solicitações recebidas, ele registra o nome do serviço do aplicativo no [campo de nome](#) do segmento.

O X-Ray SDK pode nomear segmentos com o nome do host no cabeçalho da solicitação HTTP. No entanto, esse cabeçalho pode ser falsificado, o que pode resultar em nós inesperados no mapa de serviço. Para evitar que o SDK nomeie segmentos incorretamente devido a solicitações com cabeçalhos de host falsificados, você deve especificar um nome padrão para as solicitações recebidas.

Se o aplicativo atende a solicitações para vários domínios, você pode configurar o SDK para usar uma estratégia de nomeação dinâmica para refletir isso nos nomes dos segmentos. Uma estratégia de nomeação dinâmica permite que o SDK use o nome do host para solicitações que correspondam a um padrão esperado e aplique o nome padrão às solicitações que não correspondem.

Por exemplo, você pode ter um único aplicativo para atender a solicitações para três subdomínios: `www.example.com`, `api.example.com` e `static.example.com`. Você pode usar uma estratégia de nomeação dinâmica com o padrão `*.example.com` a fim de identificar segmentos para cada subdomínio com um nome diferente, o que resulta em três nós de serviço no mapa de serviços. Se a aplicação receber solicitações com um nome de host que não corresponda ao padrão, você verá um quarto nó no mapa de serviços com um nome alternativo especificado por você.

Para usar o mesmo nome para todos os segmentos de solicitação, especifique o nome do seu aplicativo ao inicializar o middleware, como mostrado nas seções anteriores.

Note

Você pode sobrepor o nome do serviço padrão que definiu no código com a [variável de ambiente](#) `AWS_XRAY_TRACING_NAME`.

Uma estratégia de nomenclatura dinâmica define um padrão que os nomes de host devem corresponder, e um nome padrão a ser usado se o nome do host na solicitação HTTP não corresponder ao padrão. Para nomear segmentos dinamicamente, use `AWSXRay.middleware.enableDynamicNaming`.

Example app.js – nomes de segmentos dinâmicos

Se o nome do host na solicitação estiver em conformidade com o padrão `*.example.com`, use o nome do host. Caso contrário, use `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());
```

Rastreando chamadas AWS do SDK com o X-Ray SDK para Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando seu aplicativo faz chamadas Serviços da AWS para armazenar dados, gravar em uma fila ou enviar notificações, o X-Ray SDK para Node.js rastreia as chamadas downstream em subsegmentos.

O rastreamento Serviços da AWS e os recursos que você acessa nesses serviços (por exemplo, um bucket do Amazon S3 ou uma fila do Amazon SQS) aparecem como nós downstream no mapa de rastreamento no console X-Ray.

Cientes AWS do Instrument SDK que você cria por meio da AWS SDK para JavaScript V2 ou AWS SDK para JavaScript V3. Cada versão do AWS SDK fornece métodos diferentes para instrumentar clientes AWS SDK.

Note

Atualmente, o AWS X-Ray SDK para Node.js retorna menos informações de segmentos ao instrumentar clientes AWS SDK para JavaScript V3, em comparação com a instrumentação de clientes V2. Por exemplo, os subsegmentos que representam chamadas para o DynamoDB não retornarão o nome da tabela. Se você precisar dessas informações de segmento em seus rastreamentos, considere usar o AWS SDK para JavaScript V2.

AWS SDK para JavaScript V2

Você pode instrumentar todos os clientes do AWS SDK V2 agrupando sua instrução `aws-sdk` `require` em uma chamada para `AWSXRay.captureAWS`

Example app.js: instrumentação de SDK da AWS

```
const AWS = AWSXRay.captureAWS(require('aws-sdk'));
```

Para instrumentar clientes individuais, envolva seu cliente AWS SDK em uma chamada para `AWSXRay.captureAWSClient`. Por exemplo, para instrumentar um cliente AmazonDynamoDB:

Example app.js: instrumentação de cliente do DynamoDB

```
const AWSXRay = require('aws-xray-sdk');  
...  
const ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
```



Warning

Não use `captureAWS` e `captureAWSClient` juntos. Isso resultará em subsegmentos duplicados.

Se você quiser usar [TypeScript](#) com [ECMAScript módulos](#) (ESM) para carregar seu JavaScript código, use o exemplo a seguir para importar bibliotecas:

Example app.js - Instrumentação AWS do SDK

```
import * as AWS from 'aws-sdk';  
import * as AWSXRay from 'aws-xray-sdk';
```

Para instrumentar todos os AWS clientes com ESM, use o seguinte código:

Example app.js - Instrumentação AWS do SDK

```
import * as AWS from 'aws-sdk';  
import * as AWSXRay from 'aws-xray-sdk';  
const XRAY_AWS = AWSXRay.captureAWS(AWS);  
const ddb = new XRAY_AWS.DynamoDB();
```

Para todos os serviços, o nome da API chamada no console do X-Ray pode ser visto. Para um subconjunto de serviços, o X-Ray SDK adiciona informações ao segmento para fornecer maior detalhamento no mapa de serviços.

Por exemplo, quando você faz uma chamada com um cliente instrumentado do DynamoDB, o SDK adiciona o nome da tabela ao segmento para chamadas direcionadas a uma tabela. No console, cada tabela aparece como um nó separado no mapa de serviços, com um nó genérico do DynamoDB para chamadas não direcionadas a uma tabela.

Example Subsegmento para uma chamada ao DynamoDB para salvar um item

```
{
  "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": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Ao acessar recursos nomeados, as chamadas para os serviços os seguir criam nós adicionais no mapa de serviço. As chamadas que não apontam para recursos específicos criam um nó genérico para o serviço.

- Amazon DynamoDB: nome da tabela
- Amazon Simple Storage Service: nome de chave e bucket
- Amazon Simple Queue Service: nome da fila

AWS SDK para JavaScript V3

A AWS SDK para JavaScript V3 é modular, então seu código carrega apenas os módulos necessários. Por esse motivo, não é possível instrumentar todos os clientes do AWS SDK, pois a V3 não oferece suporte ao `captureAWS` método.

Se quiser usar TypeScript com ECMAScript Módulos (ESM) para carregar seu JavaScript código, você pode usar o exemplo a seguir para importar bibliotecas:

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

Instrumente cada cliente AWS SDK usando o `AWSXRay.captureAWSv3Client` método. Por exemplo, para instrumentar um cliente AmazonDynamoDB:

Example app.js: instrumentação de cliente do DynamoDB usando o SDK para Javascript V3

```
const AWSXRay = require('aws-xray-sdk');
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
...
const ddb = AWSXRay.captureAWSv3Client(new DynamoDBClient({ region:
  "region" }));
```

Ao usar a AWS SDK para JavaScript V3, metadados como nome da tabela, nome do bucket e da chave ou nome da fila não são retornados atualmente e, portanto, o mapa de rastreamento não conterá nós discretos para cada recurso nomeado, como faria ao instrumentar clientes AWS SDK usando a V2. AWS SDK para JavaScript

Example Subsegmento para uma chamada ao DynamoDB para salvar um item, ao usar a V3 AWS SDK para JavaScript

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "operation": "UpdateItem",
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

```
}
```

Rastrear chamadas para serviços da web HTTP subsequentes usando o X-Ray SDK para Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando seu aplicativo faz chamadas para microsserviços ou HTTP público APIs, você pode usar o cliente X-Ray SDK for Node.js para instrumentar essas chamadas e adicionar a API ao gráfico de serviços como um serviço downstream.

Passa o cliente `http` ou `https` para o método `captureHTTP`s do X-Ray SDK para Node.js a fim de rastrear chamadas de saída.

Note

As chamadas que usam bibliotecas de solicitação HTTP de terceiros, como Axios ou Superagent, são compatíveis com a [API `captureHTTP`sGlobal\(\)](#) e ainda serão rastreadas quando usarem o módulo nativo `http`.

Example app.js – Cliente HTTP

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTP(require('http'));
```

Para habilitar o rastreamento em todos os clientes HTTP, chame `captureHTTP`sGlobal antes de carregar `http`.

Example app.js – Cliente HTTP (global)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

Quando você instrumenta uma chamada para uma API subsequente da web, o X-Ray SDK para Node.js registra um subsegmento que contém informações sobre a solicitação e a resposta HTTP. O X-Ray usa o subsegmento para gerar um segmento inferido para a API remota.

Example Subsegmento para uma chamada HTTP downstream

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

Example Segmento inferido para uma chamada HTTP de downstream

```
{
  "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
}
```

Rastrear consultas SQL com o X-Ray SDK para Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Instrumente consultas do banco de dados SQL integrando o cliente SQL ao método de cliente correspondente do X-Ray SDK para Node.js.

- 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);
```

Quando você usa um cliente instrumentado para fazer consultas SQL, o SDK de X-Ray para Node.js registra informações sobre a conexão e a consulta em um subsegmento.

Incluir dados adicionais em subsegmentos SQL

Você pode adicionar outras informações a subsegmentos gerados para consultas SQL, desde que sejam mapeados para um campo SQL na lista de permissões. Por exemplo, para registrar a sequência de consulta SQL limpa em um subsegmento, você pode adicioná-la diretamente ao objeto SQL do subsegmento.

Example Atribuir SQL a um subsegmento

```
const queryString = 'SELECT * FROM MyTable';
connection.query(queryString, ...);

// Retrieve the most recently created subsegment
const subs = AWSXRay.getSegment().subsegments;

if (subs && subs.length > 0) {
  var sqlSub = subs[subs.length - 1];
  sqlSub.sql.sanitized_query = queryString;
}
```

Para obter uma lista completa de campos SQL na lista de permissões, consulte [Consultas SQL](#) no Guia do desenvolvedor do AWS X-Ray .

Gerar subsegmentos personalizados com o X-Ray SDK para Node.js

Os subsegmentos estendem o [segmento](#) de um rastreamento com detalhes sobre o trabalho realizado para atender a uma solicitação. Sempre que você faz uma chamada com um cliente instrumentado, o X-Ray SDK registra as informações geradas em um subsegmento. Você pode criar subsegmentos adicionais para agrupar outros subsegmentos, medir o desempenho de uma seção do código ou registrar anotações e metadados.

Subsegmentos Express personalizados

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-

Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Para criar um subsegmento personalizado para uma função que faz chamadas para serviços de downstream, use a função `captureAsyncFunc`.

Example `app.js`: subsegmentos personalizados expressos

```
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();
};
```

Neste exemplo, o aplicativo cria um subsegmento personalizado denominado `send` para chamadas para a função `sendRequest`. `captureAsyncFunc` passa um subsegmento, que você deve fechar dentro da função de retorno de chamada, quando as chamadas assíncronas que ele faz são concluídas.

Para funções síncronas, você pode usar a função `captureFunc`, que fecha o subsegmento automaticamente assim que a função bloquear finaliza a execução.

Quando você cria um subsegmento dentro de um segmento ou outro subsegmento, o X-Ray SDK para Node.js; gera um ID para ele e registra a hora de início e de término.

Example Subsegmento com metadados

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  }
},
```

Subsegmentos personalizados do Lambda

O SDK é configurado para criar automaticamente um segmento de fachada de espaço reservado quando ele detecta que está sendo executado no Lambda. Para criar um subsegmento básico, que criará um único nó `AWS::Lambda::Function` no mapa de rastreamento do X-Ray, chame e reutilize o segmento de fachada. Se você criar manualmente um novo segmento com um novo ID (ao compartilhar o ID de rastreamento, o ID pai e a decisão de amostragem), será possível enviar um novo segmento.

Example app.js: subsegmentos personalizados manuais

```
const segment = AWSXRay.getSegment(); //returns the facade segment
const subsegment = segment.addNewSubsegment('subseg');
...
subsegment.close();
//the segment is closed by the SDK automatically
```

Adicione anotações e metadados aos segmentos com o X-Ray SDK para Node.js

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode registrar informações adicionais sobre solicitações, o ambiente ou o aplicativo com anotações e metadados. Também é possível adicionar anotações e metadados aos segmentos que o X-Ray SDK cria ou aos subsegmentos personalizados que você cria.

Anotações são pares de chave-valor com valores booleanos, de string ou número. As anotações são indexadas para serem usadas com [expressões de filtro](#). Use anotações para registrar dados que você deseja usar para agrupar rastreamentos no console ou ao chamar a API [GetTraceSummaries](#).

Metadados são pares chave-valor que podem ter valores de qualquer tipo, incluindo objetos e listas, mas não são indexados para uso com expressões de filtro. Use metadados para registrar dados adicionais que você deseja armazenar no rastreamento e não precisa usar com a pesquisa.

Além de anotações e metadados, você também pode [registrar strings de ID de usuário](#) em segmentos. IDs Os usuários são registrados em um campo separado nos segmentos e são indexados para uso com a pesquisa.

Seções

- [Registrar anotações com o X-Ray SDK para Node.js](#)
- [Registrar metadados com o X-Ray SDK para Node.js](#)
- [Gravando usuário IDs com o X-Ray SDK para Node.js](#)

Registrar anotações com o X-Ray SDK para Node.js

Use anotações para registrar informações em segmentos ou subsegmentos que você deseja indexar para pesquisa.

Requisitos de anotação

- Chaves: a chave para uma anotação do X-Ray pode ter até 500 caracteres alfanuméricos. Você não pode usar espaços ou símbolos que não sejam um ponto (.)
- Valores: o valor de uma anotação do X-Ray pode ter até 1.000 caracteres Unicode.
- O número de anotações: você pode usar até cinquenta anotações por rastreamento.

Como registrar anotações

1. Obtenha uma referência para o segmento ou subsegmento.

```
var AWSXRay = require('aws-xray-sdk');  
...  
var document = AWSXRay.getSegment();
```

2. Chame `addAnnotation` com uma chave de string e um valor de número, string ou booleano.

```
document.addAnnotation("mykey", "my value");
```

O exemplo a seguir mostra como chamar `putAnnotation` com uma chave String que inclui um ponto e um valor booleano, numérico ou de string.

```
document.putAnnotation("testkey.test", "my value");
```

O SDK registra anotações como pares de chave-valor em um objeto `annotations` no documento de segmentos. Chamar `addAnnotation` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Para encontrar rastreamentos que têm anotações com valores específicos, use a palavra-chave `annotation[key]` em uma [expressão de filtro](#).

Example `app.js`: anotações

```
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) {
    ...
  });
});
```

Registrar metadados com o X-Ray SDK para Node.js

Use metadados para registrar informações em segmentos ou subsegmentos dos quais você não precisa indexados para pesquisa. Valores de metadados podem ser strings, números, booleanos ou outro objeto que possa ser serializado em uma matriz ou objeto JSON.

Como registrar metadados

1. Obtenha uma referência para o segmento ou subsegmento.

```
var AWSXRay = require('aws-xray-sdk');
```

```
...  
var document = AWSXRay.getSegment();
```

2. Chame `addMetadata` com uma chave de string, um booleano, um número, uma string ou valor de objeto e um namespace de string.

```
document.addMetadata("my key", "my value", "my namespace");
```

or

Chame `addMetadata` com apenas uma chave e valor.

```
document.addMetadata("my key", "my value");
```

Se você não especificar um namespace, o SDK usará `default`. Chamar `addMetadata` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Gravando usuário IDs com o X-Ray SDK para Node.js

Registre o usuário IDs nos segmentos da solicitação para identificar o usuário que enviou a solicitação. Essa operação não é compatível com AWS Lambda funções porque os segmentos em ambientes Lambda são imutáveis. A chamada `setUser` pode ser aplicada somente a segmentos, não a subsegmentos.

Para gravar usuário IDs

1. Obtenha uma referência para o segmento ou subsegmento.

```
var AWSXRay = require('aws-xray-sdk');  
...  
var document = AWSXRay.getSegment();
```

2. Chame `setUser()` com um ID de string do usuário que enviou a solicitação.

```
var user = 'john123';  
  
AWSXRay.getSegment().setUser(user);
```

É possível chamar `setUser` para registrar o ID de usuário assim que o aplicativo Express iniciar o processamento de uma solicitação. Se pretende usar o segmento somente para definir o ID de usuário, é possível vincular as chamadas em uma única linha.

Example app.js: ID de usuário

```
var AWS = require('aws-sdk');
var AWSXRay = require('aws-xray-sdk');
var uuidv4 = require('uuid/v4');
var ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
...
app.post('/signup', function(req, res) {
  var userId = uuidv4();
  var item = {
    'userId': {'S': userId},
    'email': {'S': req.body.email},
    'name': {'S': req.body.name}
  };

  var seg = AWSXRay.getSegment().setUser(userId);

  ddb.putItem({
    'TableName': ddbTable,
    'Item': item,
    'Expected': { email: { Exists: false } }
  }, function(err, data) {
    ...
  });
});
```

Para encontrar rastreamentos para um ID de usuário, use a `user` palavra-chave em uma [expressão de filtragem](#).

Como trabalhar com Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Há duas maneiras de instrumentar um aplicativo Python para enviar rastreamentos ao X-Ray:

- [AWS Distro for OpenTelemetry Python](#) — AWS Uma distribuição que fornece um conjunto de bibliotecas de código aberto para enviar métricas e rastreamentos correlacionados para AWS várias soluções de monitoramento, incluindo Amazon e CloudWatch OpenSearch Amazon Service AWS X-Ray, por meio [AWS do](#) Distro for Collector. OpenTelemetry
- [AWS X-Ray SDK para Python](#) — Um conjunto de bibliotecas para gerar e enviar rastreamentos para o X-Ray por meio do daemon [X-Ray](#).

Para obter mais informações, consulte [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#).

AWS Distro para OpenTelemetry Python

Com o AWS Distro para OpenTelemetry (ADOT) Python, você pode instrumentar as aplicações uma vez e enviar métricas e rastreamentos correlacionados para várias soluções de monitoramento da AWS, como o Amazon CloudWatch, o AWS X-Ray e o Amazon OpenSearch Service. O uso do X-Ray com o ADOT requer dois componentes: um SDK do OpenTelemetry habilitado para uso com o X-Ray e o coletor AWS Distro para OpenTelemetry habilitado para uso com o X-Ray. O ADOT Python comporta instrumentação automática, permitindo que a aplicação envie rastreamentos sem alterações no código.

Para começar, consulte a documentação do [AWS Distro para OpenTelemetry Python](#).

Para obter mais informações sobre como usar o AWS Distro para OpenTelemetry com o AWS X-Ray e outros Serviços da AWS, consulte [AWS Distro para OpenTelemetry](#) ou a [documentação do AWS Distro para OpenTelemetry](#).

Para obter mais informações sobre compatibilidade de idiomas e uso, consulte [AWSobservability no GitHub](#).

AWS X-Ray SDK para Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

O X-Ray SDK para Python é uma biblioteca para aplicações web do Python que fornece classes e métodos para gerar e enviar dados de rastreamento ao daemon do X-Ray. Os dados de rastreamento incluem informações sobre solicitações HTTP recebidas pelo aplicativo e chamadas que o aplicativo faz para serviços downstream usando o AWS SDK, clientes HTTP ou um conector de banco de dados SQL. Você também pode criar segmentos manualmente e adicionar informações de depuração em anotações e metadados.

Você pode baixar o SDK com o `pip`.

```
$ pip install aws-xray-sdk
```

Note

O X-Ray SDK para Python é um projeto de código aberto. Você pode acompanhar o projeto e enviar problemas e pull requests em GitHub: github.com/aws/aws-xray-sdk-python

Se você usa o Django ou o Flask, comece [adicionando middleware do SDK ao seu aplicativo](#) para rastrear solicitações recebidas. O middleware cria um [segmento](#) para cada solicitação rastreada e conclui o segmento quando a resposta é enviada. Embora o segmento esteja aberto, você pode usar os métodos do cliente do SDK para adicionar informações ao segmento e criar subsegmentos para rastrear as chamadas subsequentes. O SDK também registra automaticamente exceções que seu aplicativo lança enquanto o segmento está aberto. Para outros aplicativos, você pode [criar segmentos manualmente](#).

Para funções do Lambda chamadas por uma aplicação ou um serviço instrumentado, o Lambda lê o [cabeçalho de rastreamento](#) e rastreia automaticamente as solicitações amostradas. Para outras funções, você pode [configurar o Lambda](#) para amostrar e rastrear solicitações recebidas. Em ambos os casos, o Lambda cria o segmento e o fornece ao X-Ray SDK.

Note

No Lambda, o X-Ray SDK é opcional. Se você não o usar em sua função, mesmo assim o mapa de serviço incluirá um nó para o serviço Lambda e um para cada função do Lambda. Ao adicionar o SDK, você pode instrumentar o código da função para adicionar subsegmentos ao segmento de função registrado pelo Lambda. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Consulte [Operador](#) para ver um exemplo de função do Python instrumentada no Lambda.

Em seguida, use o X-Ray SDK para Python para instrumentar chamadas subsequentes [aplicando patches às bibliotecas da aplicação](#). O SDK é compatível com as seguintes bibliotecas.

Bibliotecas com suporte

- [botocore](#), [boto3](#) — AWS SDK for Python (Boto) Clientes de instrumentos.
- [pynamodb](#): instrumentar a versão do PynamoDB de clientes do Amazon DynamoDB.
- [aiobotocore](#), [aioboto3](#): instrumentar as versões integradas ao [asyncio](#) de clientes do SDK para Python.
- [requests](#), [aiohttp](#): instrumentar clientes HTTP de nível superior.
- [httplib](#), [http.client](#): instrumentar clientes HTTP de nível inferior e as bibliotecas de nível superior que os utilizam.
- [sqlite3](#)— SQLite Clientes de instrumentos.

- [mysql-connector-python](#): instrumentar clientes do MySQL.
- [pg8000](#): instrumentar a interface PostgreSQL Pure-Python.
- [psycopg2](#): instrumentar o adaptador de banco de dados PostgreSQL.
- [pymongo](#): instrumentar clientes do MongoDB.
- [pymysql](#)— Instrumente clientes baseados em PyMy SQL para MySQL e MariaDB.

Sempre que seu aplicativo faz chamadas para AWS um banco de dados SQL ou outros serviços HTTP, o SDK registra as informações sobre a chamada em um subsegmento. Serviços da AWS e os recursos que você acessa nos serviços aparecem como nós downstream no mapa de rastreamento para ajudá-lo a identificar erros e problemas de limitação em conexões individuais.

Depois que você começar a usar o SDK, personalize o comportamento dele [configurando o gravador e o middleware](#). Você pode adicionar plug-ins para registrar dados sobre os recursos de computação que executam seu aplicativo, personalizar o comportamento de amostragem, estipulando regras de amostragem, e definir o nível de log para ver mais ou menos informações do SDK nos logs do aplicativo.

Registre informações adicionais sobre as solicitações e o trabalho que o aplicativo faz em [anotações e metadados](#). Anotações são simples pares de chave-valor que são indexados para serem usados com [expressões de filtro](#) para que você possa pesquisar rastreamentos que contêm dados específicos. As entradas de metadados são menos restritivas e podem registrar matrizes e objetos inteiros: tudo o que pode ser serializado em JSON.

Anotações e metadados

Anotações e metadados são textos arbitrários que você adiciona aos segmentos com o X-Ray SDK. As anotações são indexadas para serem usadas com expressões de filtro. Os metadados não são indexados, mas podem ser visualizados no segmento bruto com o console ou a API do X-Ray. Qualquer pessoa à qual você conceder acesso de leitura ao X-Ray poderá visualizar esses dados.

Quando há uma grande quantidade de clientes instrumentados no código, um único segmento de solicitação pode conter um grande número de subsegmentos, um para cada chamada feita com um cliente instrumentado. Você pode organizar e agrupar subsegmentos integrando chamadas de clientes em [subsegmentos personalizados](#). Você pode criar um subsegmento personalizado para

uma função inteira ou qualquer seção do código. Você pode, então, gravar metadados e anotações no subsegmento em vez de gravar tudo no segmento pai.

Para documentação de referência para as classes e métodos do SDK, consulte a [Referência de API do AWS X-Ray X-Ray SDK para Python](#).

Requisitos

O X-Ray SDK para Python é compatível com as versões de linguagem e biblioteca a seguir.

- Python: 2.7, 3.4 e mais recente
- Django: 1.10 e mais recente
- Flask: 0.10 e mais recente
- aiohttp: 2.3.0 e mais recente
- AWS SDK for Python (Boto): 1.4.0 e mais recente
- botocore: 1.5.0 e mais recente
- enum: 0.4.7 e mais recente, para Python versões 3.4.0 e mais antigas
- jsonpickle: 1.0.0 e mais recente
- setuptools: 40.6.3 e mais recente
- wrapt: 1.11.0 e mais recente

Gerenciamento de dependências

O X-Ray SDK para Python está disponível em `pip`.

- Pacote: `aws-xray-sdk`

Adicione o SDK como uma dependência em seu arquivo `requirements.txt`.

Example requirements.txt

```
aws-xray-sdk==2.4.2
boto3==1.4.4
botocore==1.5.55
Django==1.11.3
```

Se você usa o Elastic Beanstalk para implantar a aplicação, todos os pacotes são instalados automaticamente em `requirements.txt`.

Configurar o X-Ray SDK para Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

O X-Ray SDK para Python tem uma classe chamada `xray_recorder`, que fornece o gravador global. Você pode configurar o gravador global para personalizar o middleware que cria segmentos para chamadas HTTP de entrada.

Seções

- [Plug-ins de serviço](#)
- [Regras de amostragem](#)
- [Registro em log](#)
- [Configuração do gravador no código](#)
- [Configuração do gravador com o Django](#)
- [Variáveis de ambiente](#)

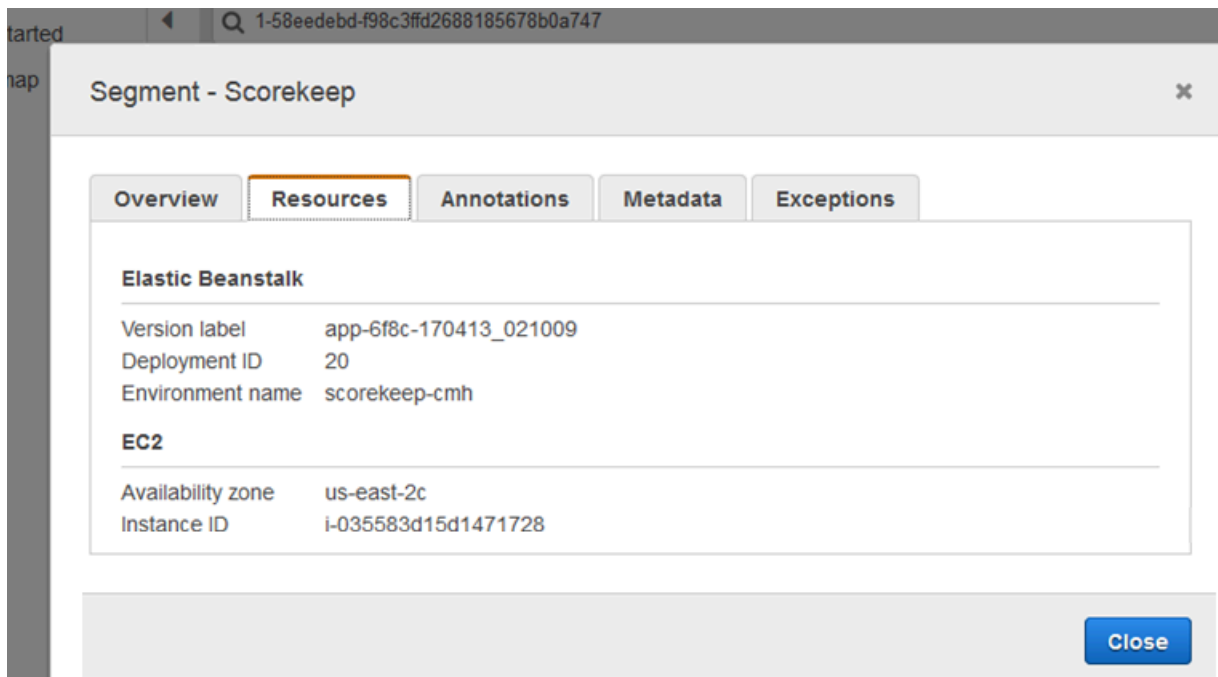
Plug-ins de serviço

Use plugins para registrar informações sobre o serviço que hospeda o aplicativo.

Plugins

- Amazon EC2 — `EC2Plugin` adiciona o ID da instância, a zona de disponibilidade e o grupo de CloudWatch registros.

- Elastic Beanstalk: o `ElasticBeanstalkPlugin` adiciona o nome do ambiente, o rótulo da versão e o ID de implantação.
- Amazon ECS: o `EC2Plugin` adiciona o ID do contêiner.



Para usar um plug-in, chame configure no `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

xray_recorder.configure(service='My app')
plugins = ('ElasticBeanstalkPlugin', 'EC2Plugin')
xray_recorder.configure(plugins=plugins)
patch_all()
```

Note

Como os plugins são passados como uma tupla, inclua uma `,` à direita ao especificar um único plug-in. Por exemplo, `plugins = ('EC2Plugin',)`.

Você também pode usar [variáveis de ambiente](#), que têm precedência sobre valores definidos no código, para configurar o gravador.

Configure plug-ins antes de [bibliotecas de patches](#) para gravar chamadas downstream.

O SDK também usa as configurações do plug-in para definir o campo `origin` no segmento. Isso indica o tipo de AWS recurso que executa seu aplicativo. Quando você usa vários plug-ins, o SDK usa a seguinte ordem de resolução para determinar a origem: ElasticBeanstalk > EKS > ECS > EC2

Regras de amostragem

O SDK usa as regras de amostragem que você define no console do X-Ray para determinar quais solicitações serão registradas. A regra padrão rastreia a primeira solicitação a cada segundo e 5% de todas as solicitações adicionais em todos os serviços que enviam rastreamentos ao X-Ray. [Crie regras adicionais no console do X-Ray](#) para personalizar a quantidade de dados registrados para cada uma dos aplicativos.

O SDK aplica regras personalizadas na ordem em que elas estão definidas. Se uma solicitação corresponder a várias regras personalizadas, o SDK aplicará somente a primeira regra.

Note

Se o SDK não conseguir acessar o X-Ray para obter regras de amostragem, ele reverterá para uma regra local padrão da primeira solicitação a cada segundo e 5% de todas as solicitações adicionais por host. Isso pode ocorrer se o host não tiver permissão para chamar a amostragem APIs ou não conseguir se conectar ao daemon X-Ray, que atua como um proxy TCP para chamadas de API feitas pelo SDK.

Você também pode configurar o SDK para carregar regras de amostragem de um documento JSON. O SDK pode usar regras locais como backup para casos em que a amostragem do X-Ray não está disponível ou usar exclusivamente regras locais.

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
}
}
```

Este exemplo define uma regra personalizada e uma regra padrão. A regra personalizada aplica uma taxa de amostragem de 5% sem um número mínimo de solicitações para rastrear os caminhos em `/api/move/`. A regra padrão rastreia a primeira solicitação a cada segundo e 10% das solicitações adicionais.

A desvantagem de definir regras localmente é que o destino fixo é aplicado por instância do gravador de forma independente, em vez de ser gerenciado pelo serviço X-Ray. À medida que você implanta mais hosts, a taxa fixa é multiplicada, dificultando o controle da quantidade de dados registrados.

Ativado AWS Lambda, você não pode modificar a taxa de amostragem. Se sua função for chamada por um serviço instrumentado, as chamadas que geraram solicitações que foram amostradas por esse serviço serão registradas pelo Lambda. Se o rastreamento ativo estiver habilitado e nenhum cabeçalho de rastreamento estiver presente, o Lambda tomará a decisão de amostragem.

Para configurar as regras de amostragem de `backupxray_recorder.configure`, chame, conforme mostrado no exemplo a seguir, where *rules* é um dicionário de regras ou o caminho absoluto para um arquivo JSON contendo regras de amostragem.

```
xray_recorder.configure(sampling_rules=rules)
```

Para usar somente regras locais, configure o gravador com um `LocalSampler`.

```
from aws_xray_sdk.core.sampling.local.sampler import LocalSampler
xray_recorder.configure(sampler=LocalSampler())
```

Você também pode configurar o gravador global para desabilitar a amostragem e instrumentar todas as solicitações de entrada.

Example main.py: desabilitar a amostragem

```
xray_recorder.configure(sampling=False)
```

Registro em log

O SDK usa o módulo `logging` integrado do Python com um nível de registro em log `WARNING` padrão. Obtenha uma referência ao agente de log para a classe `aws_xray_sdk` e chame `setLevel` nela para configurar o nível de log diferente para a biblioteca e o restante de seu aplicativo.

Example app.py: Registrar em log

```
logging.basicConfig(level='WARNING')
logging.getLogger('aws_xray_sdk').setLevel(logging.ERROR)
```

Use logs de depuração para identificar problemas como subsegmentos não fechados ao [gerar subsegmentos manualmente](#).

Configuração do gravador no código

Configurações adicionais estão disponíveis no método `configure` no `xray_recorder`.

- `context_missing`: defina como `LOG_ERROR` para evitar o lançamento de exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.
- `daemon_address`: defina o host e a porta do receptor do daemon do X-Ray.
- `service`: defina um nome de serviço para o SDK usar para segmentos.
- `plugins`— Registre informações sobre os AWS recursos do seu aplicativo.
- `sampling`: defina como `False` para desabilitar a amostragem.
- `sampling_rules`: defina o caminho do arquivo JSON que contém suas [regras de amostragem](#).

Example main.py: desabilitar exceções de contexto ausente

```
from aws_xray_sdk.core import xray_recorder

xray_recorder.configure(context_missing='LOG_ERROR')
```

Configuração do gravador com o Django

Se você usar a estrutura do Django, poderá usar o arquivo `settings.py` do Django para configurar as opções no gravador global.

- `AUTO_INSTRUMENT` (Somente para Django): registre os subsegmentos para operações de renderização de banco de dados e de modelo integrados.
- `AWS_XRAY_CONTEXT_MISSING`: defina como `LOG_ERROR` para evitar o lançamento de exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.
- `AWS_XRAY_DAEMON_ADDRESS`: defina o host e a porta do receptor do daemon do X-Ray.
- `AWS_XRAY_TRACING_NAME`: defina um nome de serviço para o SDK usar para segmentos.
- `PLUGINS`— Registre informações sobre os AWS recursos do seu aplicativo.
- `SAMPLING`: defina como `False` para desabilitar a amostragem.
- `SAMPLING_RULES`: defina o caminho do arquivo JSON que contém suas [regras de amostragem](#).

Para habilitar a configuração do gravador em `settings.py`, adicione o middleware do Django à lista de aplicativos instalados.

Example `settings.py`: aplicações instalados

```
INSTALLED_APPS = [  
    ...  
    'django.contrib.sessions',  
    'aws_xray_sdk.ext.django',  
]
```

Configurar as configurações disponíveis em um dict chamado `XRAY_RECORDER`.

Example `settings.py`: aplicações instalados

```
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', 'ECSPPlugin'),  
    'SAMPLING': False,  
}
```

Variáveis de ambiente

É possível usar variáveis de ambiente para configurar o X-Ray SDK para Python. O SDK é compatível com as seguintes variáveis:

- `AWS_XRAY_TRACING_NAME`: defina um nome de serviço para o SDK usar para segmentos. Substitui o nome do serviço definido programaticamente.
- `AWS_XRAY_SDK_ENABLED`: quando definido como `false`, desabilita o SDK. Por padrão, o SDK está habilitado, a menos que a variável de ambiente esteja definida como falsa.
 - Quando desabilitado, o gravador global gera automaticamente segmentos e subsegmentos fictícios que não são enviados para o daemon, e a aplicação automática de patches está desabilitada. Os middlewares são gravados como um wrapper pelo gravador global. Toda a geração de segmentos e subsegmentos pelo middleware também se torna segmentos fictícios e subsegmentos fictícios.
- Defina o valor de `AWS_XRAY_SDK_ENABLED` por meio da variável de ambiente ou por meio de interação direta com o objeto `global_sdk_config` da biblioteca de `aws_xray_sdk`. As configurações para a variável de ambiente substituem essas interações.
- `AWS_XRAY_DAEMON_ADDRESS`: defina o host e a porta do receptor do daemon do X-Ray. Por padrão, o SDK usa `127.0.0.1:2000` para dados de rastreamento (UDP) e para amostragem (TCP). Use essa variável se você tiver configurado o daemon para [escutar em uma porta diferente](#) ou se ele estiver sendo executado em um host diferente.

Formato

- Mesma porta: `address:port`
- Portas diferentes: `tcp:address:port` `udp:address:port`
- `AWS_XRAY_CONTEXT_MISSING`: defina como `RUNTIME_ERROR` para lançar exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.

Valores válidos

- `RUNTIME_ERROR`: lance uma exceção de tempo de execução.
- `LOG_ERROR`: registre um erro e continue (padrão).
- `IGNORE_ERROR`: ignore o erro e continue.

Erros relativos a segmentos ou subsegmentos ausentes poderão ocorrer quando você tentar usar um cliente instrumentado no código de inicialização que é executado quando nenhuma solicitação estiver aberta ou em um código que gere um novo thread.

As variáveis de ambiente substituem os valores definidos no código.

Rastrear solicitações de entrada com o middleware do X-Ray SDK para Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando você adiciona o middleware ao aplicativo e configura o nome do segmento, o X-Ray SDK para Python cria um segmento para cada solicitação amostrada. Este segmento inclui tempo, método e disposição da solicitação HTTP. Instrumentação adicional cria subsegmentos sobre esse segmento.

O X-Ray SDK para Python é compatível com o seguinte middleware para instrumentar as solicitações HTTP de entrada:

- Django
- Flask
- Bottle

Note

Para AWS Lambda funções, o Lambda cria um segmento para cada solicitação de amostra. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Consulte [Operador](#) para ver um exemplo de função do Python instrumentada no Lambda.

Para scripts ou aplicativos Python em outros frameworks, você pode [criar segmentos manualmente](#).

Cada segmento tem um nome que identifica a aplicação no mapa de serviços. O segmento pode ser nomeado estaticamente ou você pode configurar o SDK para nomeá-lo dinamicamente com

base no cabeçalho do host na solicitação de entrada. A nomeação dinâmica permite agrupar rastreamentos com base no nome de domínio na solicitação e aplicar um nome padrão se o nome não corresponder ao padrão esperado (por exemplo, se o cabeçalho do host for falsificado).

Solicitações encaminhadas

Se um balanceador de carga ou outro intermediário encaminhar uma solicitação para o aplicativo, o X-Ray obterá o IP do cliente do cabeçalho `X-Forwarded-For` na solicitação em vez do IP de origem no pacote IP. O IP do cliente registrado para uma solicitação encaminhada pode ser falsificado; portanto, não é digno de confiança.

Quando uma solicitação é encaminhada, o SDK define um campo adicional no segmento para indicar isso. Se o segmento tiver o campo `x_forwarded_for` definido como `true`, isso significa que o IP do cliente foi obtido no cabeçalho `X-Forwarded-For` na solicitação HTTP.

O middleware cria um segmento para cada solicitação de entrada com um bloco `http` contendo as seguintes informações:

- Método HTTP: GET, POST, PUT, DELETE etc.
- Endereço do cliente: o endereço IP do cliente que enviou a solicitação.
- Código de resposta: o código de resposta HTTP da solicitação concluída.
- Horário: a hora de início (quando a solicitação foi recebida) e a hora de término (quando a resposta foi enviada).
- Agente do usuário: o `user-agent` da solicitação.
- Tamanho do conteúdo: o `content-length` da resposta.

Seções

- [Adicionar o middleware a seu aplicativo \(Django\)](#)
- [Adicionar o middleware a seu aplicativo \(Flask\)](#)
- [Adicionar o middleware a seu aplicativo \(Bottle\)](#)
- [Instrumentar código Python manualmente](#)
- [Configurar uma estratégia de nomeação de segmentos](#)

Adicionar o middleware a seu aplicativo (Django)

Adicione o middleware à lista de MIDDLEWARE em seu arquivo `settings.py`. O middleware do X-Ray deve ser a primeira linha em seu arquivo `settings.py` para garantir que as solicitações com falha em outro middleware sejam gravadas.

Example `settings.py`: middleware do X-Ray SDK para Python

```
MIDDLEWARE = [  
    'aws_xray_sdk.ext.django.middleware.XRayMiddleware',  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware'  
]
```

Adicione a aplicação Django do X-Ray SDK à lista `INSTALLED_APPS` em seu arquivo `settings.py`. Isso permitirá que o gravador do X-Ray seja configurado durante a inicialização da aplicação.

Example `settings.py`: aplicação Django do X-Ray SDK para Python

```
INSTALLED_APPS = [  
    'aws_xray_sdk.ext.django',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

Configure um nome de segmento em seu [arquivo settings.py](#).

Example `settings.py`: nome do segmento

```
XRAY_RECORDER = {  
    'AWS_XRAY_TRACING_NAME': 'My application',  
    'PLUGINS': ('EC2Plugin',),  
}
```

```
}
```

Isso instrui o gravador do X-Ray a rastrear solicitações atendidas pela aplicação Django com a taxa de amostragem padrão. Você pode [configurar o gravador com seu arquivo de configurações do Django](#) para aplicar regras de amostragem personalizadas ou alterar outras configurações.

Note

Como os plugins são passados como uma tupla, inclua uma `,` à direita ao especificar um único plug-in. Por exemplo, `plugins = ('EC2Plugin',)`.

Adicionar o middleware a seu aplicativo (Flask)

Para instrumentar seu aplicativo Flask, primeiro configure um nome de segmento no `xray_recorder`. Em seguida, use a função `XRayMiddleware` para corrigir seu aplicativo Flask no código.

Example app.py

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.ext.flask.middleware import XRayMiddleware

app = Flask(__name__)

xray_recorder.configure(service='My application')
XRayMiddleware(app, xray_recorder)
```

Isso instrui o gravador do X-Ray a rastrear solicitações atendidas pela aplicação Flask com a taxa de amostragem padrão. Você pode [configurar o gravador em código](#) para aplicar regras de amostragem personalizadas ou alterar outras configurações.

Adicionar o middleware a seu aplicativo (Bottle)

Para instrumentar seu aplicativo Bottle, primeiro configure um nome de segmento no `xray_recorder`. Depois, use a função `XRayMiddleware` para corrigir seu aplicativo Bottle no código.

Example app.py

```
from aws_xray_sdk.core import xray_recorder
```

```
from aws_xray_sdk.ext.bottle.middleware import XRayMiddleware

app = Bottle()

xray_recorder.configure(service='fallback_name', dynamic_naming='My application')
app.install(XRayMiddleware(xray_recorder))
```

Isso instrui o gravador do X-Ray a rastrear solicitações atendidas pela aplicação Bottle com a taxa de amostragem padrão. Você pode [configurar o gravador em código](#) para aplicar regras de amostragem personalizadas ou alterar outras configurações.

Instrumentar código Python manualmente

Se você não usar o Django ou o Flask, pode criar segmentos manualmente. Você pode criar um segmento para cada solicitação recebida ou criar segmentos em torno de clientes HTTP ou AWS SDK corrigidos para fornecer contexto para que o gravador adicione subsegmentos.

Example main.py: instrumentação manual

```
from aws_xray_sdk.core import xray_recorder

# Start a segment
segment = xray_recorder.begin_segment('segment_name')
# Start a subsegment
subsegment = xray_recorder.begin_subsegment('subsegment_name')

# Add metadata and annotations
segment.put_metadata('key', dict, 'namespace')
subsegment.put_annotation('key', 'value')

# Close the subsegment and segment
xray_recorder.end_subsegment()
xray_recorder.end_segment()
```

Configurar uma estratégia de nomeação de segmentos

AWS X-Ray usa um nome de serviço para identificar seu aplicativo e diferenciá-lo dos outros aplicativos, bancos de dados APIs, AWS recursos externos e que seu aplicativo usa. Quando o X-Ray SDK gera segmentos para solicitações recebidas, ele registra o nome do serviço do aplicativo no [campo de nome](#) do segmento.

O X-Ray SDK pode nomear segmentos com o nome do host no cabeçalho da solicitação HTTP. No entanto, esse cabeçalho pode ser falsificado, o que pode resultar em nós inesperados no mapa de serviço. Para evitar que o SDK nomeie segmentos incorretamente devido a solicitações com cabeçalhos de host falsificados, você deve especificar um nome padrão para as solicitações recebidas.

Se o aplicativo atende a solicitações para vários domínios, você pode configurar o SDK para usar uma estratégia de nomeação dinâmica para refletir isso nos nomes dos segmentos. Uma estratégia de nomeação dinâmica permite que o SDK use o nome do host para solicitações que correspondam a um padrão esperado e aplique o nome padrão às solicitações que não correspondem.

Por exemplo, você pode ter um único aplicativo para atender a solicitações para três subdomínios: `www.example.com`, `api.example.com` e `static.example.com`. Você pode usar uma estratégia de nomeação dinâmica com o padrão `*.example.com` a fim de identificar segmentos para cada subdomínio com um nome diferente, o que resulta em três nós de serviço no mapa de serviços. Se a aplicação receber solicitações com um nome de host que não corresponda ao padrão, você verá um quarto nó no mapa de serviços com um nome alternativo especificado por você.

Para usar o mesmo nome para todos os segmentos de solicitação, especifique o nome da aplicação ao configurar o gravador, conforme mostrado nas [seções anteriores](#).

Uma estratégia de nomeação dinâmica define um padrão com o qual os nomes de host devem corresponder e um nome padrão a ser usado se o nome do host na solicitação HTTP não corresponder ao padrão. Para nomear segmentos dinamicamente no Django, adicione a configuração `DYNAMIC_NAMING` a seu arquivo [settings.py](#).

Example settings.py: nomeação dinâmica

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_TRACING_NAME': 'My application',
    'DYNAMIC_NAMING': '*.example.com',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin')
}
```

Você pode usar `*` no padrão para fazer a correspondência com qualquer string ou `?` para um caractere único. Para o Flask, [configure o gravador no código](#).

Example main.py: nome do segmento

```
from aws_xray_sdk.core import xray_recorder
```

```
xray_recorder.configure(service='My application')
xray_recorder.configure(dynamic_naming='*.example.com')
```

Note

Você pode sobrepor o nome do serviço padrão que definiu no código com a [variável de ambiente](#) `AWS_XRAY_TRACING_NAME`.

Aplicar patches a bibliotecas para instrumentar chamadas downstream

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Para instrumentar chamadas subsequentes, use o X-Ray SDK para Python para aplicar patches às bibliotecas usadas pela aplicação. O X-Ray SDK para Python pode corrigir as bibliotecas a seguir.

Bibliotecas com suporte

- [botocore](#), [boto3](#) — AWS SDK for Python (Boto) Clientes de instrumentos.
- [pynamodb](#): instrumentar a versão do PynamoDB de clientes do Amazon DynamoDB.
- [aiobotocore](#), [aioboto3](#): instrumentar as versões integradas ao [asyncio](#) de clientes do SDK para Python.
- [requests](#), [aiohttp](#): instrumentar clientes HTTP de nível superior.
- [httplib](#), [http.client](#): instrumentar clientes HTTP de nível inferior e as bibliotecas de nível superior que os utilizam.
- [sqlite3](#) — SQLite Clientes de instrumentos.
- [mysql-connector-python](#): instrumentar clientes do MySQL.

- [pg8000](#): instrumentar a interface PostgreSQL Pure-Python.
- [psycopg2](#): instrumentar o adaptador de banco de dados PostgreSQL.
- [pymongo](#): instrumentar clientes do MongoDB.
- [pymysql](#)— Instrumente clientes baseados em PyMy SQL para MySQL e MariaDB.

Quando você usa uma biblioteca com patches aplicados, o X-Ray SDK para Python cria um subsegmento para a chamada e grava informações da solicitação e da resposta. Um segmento deve estar disponível para o SDK para criar o subsegmento, no middleware do SDK ou no AWS Lambda.

Note

Se você usa SQLAlchemy ORM, pode instrumentar suas consultas SQL importando a versão do SDK das classes SQLAlchemy de sessão e consulta. Consulte [Usar SQLAlchemy ORM](#) para obter instruções.

Para aplicar patch a todas as bibliotecas disponíveis, use a função `patch_all` em `aws_xray_sdk.core`. Algumas bibliotecas, como `httplib` e `urllib`, podem precisar habilitar a aplicação de patch dupla chamando `patch_all(double_patch=True)`.

Example main.py: aplicar patch a todas as bibliotecas compatíveis

```
import boto3
import botocore
import requests
import sqlite3

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

patch_all()
```

Para aplicar um patch a uma única biblioteca, chame `patch` com uma tupla do nome da biblioteca. Para isso, você precisará fornecer uma única lista de elementos.

Example main.py: aplicar patch a bibliotecas específicas

```
import boto3
```



```
import boto3
import requests
import mysql-connector-python

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch

libraries = (['boto3'])
patch(libraries)
```

Note

Em alguns casos, a chave que você usa para aplicar um patch a uma biblioteca não corresponde ao nome da biblioteca. Algumas chaves servem como aliases para uma ou mais bibliotecas.

Aliases de bibliotecas

- http lib: [http lib](#) e [http.client](#)
- mysql – [mysql-connector-python](#)

Rastrear contexto para trabalhos assíncronos

Para bibliotecas integradas `asyncio` ou para [criar subsegmentos para funções assíncronas](#), você também deve configurar o X-Ray SDK para Python com um contexto assíncrono. Importe a classe `AsyncContext` e passe um exemplo dela para o gravador do X-Ray.

Note

As bibliotecas de suporte a framework da Web, como `AIOHTTP`, não são tratadas pelo módulo `aws_xray_sdk.core.patcher`. Elas não serão exibidas no catálogo de `patcher` de bibliotecas com suporte.

Example main.py: aplicar patch a aioboto3

```
import asyncio
import aioboto3
import requests
```

```
from aws_xray_sdk.core.async_context import AsyncContext
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='my_service', context=AsyncContext())
from aws_xray_sdk.core import patch

libraries = (['aioboto3'])
patch(libraries)
```

Rastreando chamadas AWS do SDK com o X-Ray SDK para Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando seu aplicativo faz chamadas Serviços da AWS para armazenar dados, gravar em uma fila ou enviar notificações, o X-Ray SDK para Python rastreia as chamadas downstream em subsegmentos.

Os recursos rastreados Serviços da AWS e que você acessa nesses serviços (por exemplo, um bucket do Amazon S3 ou uma fila do Amazon SQS) aparecem como nós downstream no mapa de rastreamento no console X-Ray.

O X-Ray SDK para Python instrumenta automaticamente AWS todos os clientes do SDK quando [você](#) corrige a biblioteca. `botocore` Você não pode instrumentar clientes individuais.

Para todos os serviços, o nome da API chamada no console do X-Ray pode ser visto. Para um subconjunto de serviços, o X-Ray SDK adiciona informações ao segmento para fornecer maior detalhamento no mapa de serviços.

Por exemplo, quando você faz uma chamada com um cliente instrumentado do DynamoDB, o SDK adiciona o nome da tabela ao segmento para chamadas direcionadas a uma tabela. No console, cada tabela aparece como um nó separado no mapa de serviços, com um nó genérico do DynamoDB para chamadas não direcionadas a uma tabela.

Example Subsegmento para uma chamada ao DynamoDB para salvar um item

```
{
  "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": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Ao acessar recursos nomeados, as chamadas para os serviços os seguir criam nós adicionais no mapa de serviço. As chamadas que não apontam para recursos específicos criam um nó genérico para o serviço.

- Amazon DynamoDB: nome da tabela
- Amazon Simple Storage Service: nome de chave e bucket
- Amazon Simple Queue Service: nome da fila

Rastrear chamadas para serviços web HTTP subsequentes usando o X-Ray SDK para Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais

informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Quando seu aplicativo faz chamadas para microsserviços ou HTTP público APIs, você pode usar o X-Ray SDK para Python para instrumentar essas chamadas e adicionar a API ao gráfico de serviços como um serviço downstream.

Para instrumentar clientes HTTP, [aplique um patch à biblioteca](#) que você usa para fazer chamadas de saída. Se você usa `requests` ou o cliente HTTP integrado Python, isso é tudo o que você precisa fazer. Para o `aiohttp`, configure também o gravador com um [contexto assíncrono](#).

Se você usar a API do cliente do `aiohttp` 3, também precisará configurar o `ClientSession` com uma instância da configuração de rastreamento fornecida pelo SDK.

Exemplo [API do cliente do `aiohttp` 3](#)

```
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()
```

Quando você instrumenta uma chamada para uma API subsequente da web, o X-Ray SDK para Python registra um subsegmento que contém informações sobre a solicitação e a resposta HTTP. O X-Ray usa o subsegmento para gerar um segmento inferido para a API remota.

Exemplo Subsegmento para uma chamada HTTP downstream

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

```
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example Segmento inferido para uma chamada HTTP de downstream

```
{
  "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
}
```

Gerar subsegmentos personalizados com o X-Ray SDK para Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Os subsegmentos estendem o [segmento](#) de um rastreamento com detalhes sobre o trabalho realizado para atender a uma solicitação. Sempre que você faz uma chamada com um cliente instrumentado, o X-Ray SDK registra as informações geradas em um subsegmento. Você pode criar subsegmentos adicionais para agrupar outros subsegmentos, medir o desempenho de uma seção do código ou registrar anotações e metadados.

Para gerenciar subsegmentos, use os métodos `begin_subsegment` e `end_subsegment`.

Example main.py: subsegmento personalizado

```
from aws_xray_sdk.core import xray_recorder

subsegment = xray_recorder.begin_subsegment('annotations')
subsegment.put_annotation('id', 12345)
xray_recorder.end_subsegment()
```

Para criar um subsegmento para uma função síncrona, use o decorador `@xray_recorder.capture`. Você pode passar um nome para o subsegmento para a função de captura ou deixá-lo usar o nome da função.

Example main.py: subsegmento de função

```
from aws_xray_sdk.core import xray_recorder

@xray_recorder.capture('## create_user')
def create_user():
    ...
```

Para uma função assíncrona, use o decorador `@xray_recorder.capture_async` e passe um contexto assíncrono para o gravador.

Example main.py: subsegmento de função assíncrona

```
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()
```

Quando você cria um subsegmento dentro de um segmento ou outro subsegmento, o X-Ray SDK para Python gera um ID para ele e registra a hora de início e de término.

Example Subsegmento com metadados

```
"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"  
    }  
  },  
},
```

Adicionar anotações e metadados aos segmentos com o X-Ray SDK para Python

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Você pode registrar informações adicionais sobre solicitações, o ambiente ou o aplicativo com anotações e metadados. Também é possível adicionar anotações e metadados aos segmentos que o X-Ray SDK cria ou aos subsegmentos personalizados que você cria.

Anotações são pares de chave-valor com valores booleanos, de string ou número. As anotações são indexadas para serem usadas com [expressões de filtro](#). Use anotações para registrar

dados que você deseja usar para agrupar rastreamentos no console ou ao chamar a API [GetTraceSummaries](#).

Metadados são pares chave-valor que podem ter valores de qualquer tipo, incluindo objetos e listas, mas não são indexados para uso com expressões de filtro. Use metadados para registrar dados adicionais que você deseja armazenar no rastreamento e não precisa usar com a pesquisa.

Além de anotações e metadados, você também pode [registrar strings de ID de usuário](#) em segmentos. IDs Os usuários são registrados em um campo separado nos segmentos e são indexados para uso com a pesquisa.

Seções

- [Registrar anotações com o X-Ray SDK para Python](#)
- [Registrar metadados com o X-Ray SDK para Python](#)
- [Gravando usuário IDs com o X-Ray SDK para Python](#)

Registrar anotações com o X-Ray SDK para Python

Use anotações para registrar informações em segmentos ou subsegmentos que você deseja indexar para pesquisa.

Requisitos de anotação

- Chaves: a chave para uma anotação do X-Ray pode ter até 500 caracteres alfanuméricos. Você não pode usar espaços ou símbolos que não sejam um ponto (.)
- Valores: o valor de uma anotação do X-Ray pode ter até 1.000 caracteres Unicode.
- O número de anotações: você pode usar até cinquenta anotações por rastreamento.

Como registrar anotações

1. Obtenha uma referência para o segmento ou subsegmento atual no `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

ou


```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. Chame `put_annotation` com uma chave de string e um valor de número, string ou booleano.

```
document.put_annotation("mykey", "my value");
```

O exemplo a seguir mostra como chamar `putAnnotation` com uma chave `String` que inclui um ponto e um valor booleano, numérico ou de string.

```
document.putAnnotation("testkey.test", "my value");
```

Também é possível usar o método `put_annotation` em `xray_recorder`. Esse método registra anotações no subsegmento atual ou, se nenhum subsegmento estiver aberto, no segmento.

```
xray_recorder.put_annotation("mykey", "my value");
```

O SDK registra anotações como pares de chave-valor em um objeto `annotations` no documento de segmentos. Chamar `put_annotation` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Para encontrar rastreamentos que têm anotações com valores específicos, use a palavra-chave `annotation[key]` em uma [expressão de filtro](#).

Registrar metadados com o X-Ray SDK para Python

Warning

Não adicione objetos com referências circulares como valores de metadados no SDK do X-Ray para Python. Esses objetos não podem ser serializados em JSON e podem criar loops infinitos no SDK. Além disso, evite adicionar objetos grandes e complexos como metadados para evitar problemas de desempenho.

Use metadados para registrar informações em segmentos ou subsegmentos dos quais você não precisa indexados para pesquisa. Valores de metadados podem ser strings, números, booleanos ou qualquer objeto que possa ser serializado em uma matriz ou objeto JSON.

Como registrar metadados

1. Obtenha uma referência para o segmento ou subsegmento atual no `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

ou

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. Chame `put_metadata` com uma chave de string, um booleano, um número, uma string ou valor de objeto e um namespace de string.

```
document.put_metadata("my key", "my value", "my namespace");
```

or

Chame `put_metadata` com apenas uma chave e valor.

```
document.put_metadata("my key", "my value");
```

Também é possível usar o método `put_metadata` em `xray_recorder`. Esse método registra metadados no subsegmento atual ou, se nenhum subsegmento estiver aberto, no segmento.

```
xray_recorder.put_metadata("my key", "my value");
```

Se você não especificar um namespace, o SDK usará `default`. Chamar `put_metadata` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Gravando usuário IDs com o X-Ray SDK para Python

Registre o usuário IDs nos segmentos da solicitação para identificar o usuário que enviou a solicitação.

Para gravar usuário IDs

1. Obtenha uma referência para o segmento atual em `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

2. Chame `setUser` com um ID de string do usuário que enviou a solicitação.

```
document.set_user("U12345");
```

Você pode acionar `set_user` em seus controladores para registrar o ID de usuário assim que o aplicativo começar a processar uma solicitação.

Para encontrar rastreamentos para um ID de usuário, use a `user` palavra-chave em uma [expressão de filtragem](#).

Instrumentar frameworks da web implantadas em ambientes sem servidor

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O AWS X-Ray SDK para Python oferece suporte a estruturas web de instrumentação implantadas em aplicativos sem servidor. Sem servidor é a arquitetura nativa da nuvem que permite transferir mais das suas responsabilidades operacionais à AWS, aumentando a agilidade e a inovação.

Arquitetura sem servidor é um modelo de aplicativo de software que permite que você crie e execute aplicativos e serviços sem se preocupar com servidores. Ela elimina as tarefas de gerenciamento de infraestrutura, como provisionamento de servidores ou de clusters, aplicação de patches, manutenção do sistema operacional e provisionamento de capacidade. Você pode criar soluções sem servidor para praticamente qualquer tipo de aplicativo ou serviço de back-end, e cuidaremos de tudo o que for necessário para executar e escalar aplicativos com alta disponibilidade.

Este tutorial mostra como instrumentar automaticamente AWS X-Ray em uma estrutura web, como Flask ou Django, que é implantada em um ambiente sem servidor. A instrumentação X-Ray do aplicativo permite que você visualize todas as chamadas downstream feitas, começando pelo Amazon API Gateway até sua AWS Lambda função, e as chamadas de saída que seu aplicativo faz.

O X-Ray SDK para Python é compatível com os seguintes frameworks de aplicação Python:

- Flask versão 0.8 ou posterior
- Django versão 1.0 ou posterior

Este tutorial desenvolve um exemplo de aplicação sem servidor que é implantada no Lambda e invocada pelo API Gateway. Este tutorial usa o Zappa para implantar automaticamente a aplicação no Lambda e configurar o endpoint do API Gateway.

Pré-requisitos

- [Zappa](#)
- [Python](#): versão 2.7 ou 3.6.
- [AWS CLI](#)— Verifique se você AWS CLI está configurado com a conta e Região da AWS na qual você implantará seu aplicativo.
- [Pip](#)
- [Virtualenv](#)

Etapa 1: Criar um ambiente

Nesta etapa, você cria um ambiente virtual usando `virtualenv` para hospedar um aplicativo.

1. Usando o AWS CLI, crie um diretório para o aplicativo. Altere para o novo diretório.

```
mkdir serverless_application
```

```
cd serverless_application
```

2. Em seguida, crie um ambiente virtual no novo diretório. Use o comando a seguir para ativá-lo:

```
# Create our virtual environment
virtualenv serverless_env

# Activate it
source serverless_env/bin/activate
```

3. Instale o X-Ray, o Flask, o Zappa e a biblioteca de solicitações em seu ambiente.

```
# Install X-Ray, Flask, Zappa, and Requests into your environment
pip install aws-xray-sdk flask zappa requests
```

4. Adicione o código do aplicativo ao diretório `serverless_application`. Para este exemplo, podemos criar a partir do exemplo [Hello World](#) do Flask.

No diretório `serverless_application`, crie um arquivo chamado `my_app.py`. Depois, use um editor de texto para adicionar os comandos a seguir. Este aplicativo instrumenta a biblioteca de solicitações, aplica patches ao middleware do aplicativo Flask e abre o endpoint `'/'`.

```
# 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
```

Etapa 2: Criar e implantar um ambiente do Zappa

Nesta etapa, você usará o Zappa para configurar automaticamente um endpoint do API Gateway e implantá-lo no Lambda.

1. Inicialize o Zappa de dentro do diretório `serverless_application`. Para este exemplo, usamos as configurações padrão, mas se você tiver preferências de personalização, o Zappa exibirá as instruções de configuração.

```
zappa init
```

```
What do you want to call this environment (default 'dev'): dev
...
What do you want to call your bucket? (default 'zappa-*****'): zappa-*****
...
...
It looks like this is a Flask application.
What's the modular path to your app's function?
This will likely be something like 'your_module.app'.
We discovered: my_app.app
Where is your app's function? (default 'my_app.app'): my_app.app
...
Would you like to deploy this application globally? (default 'n') [y/n/
(p)rimary]: n
```

2. Habilite o X-Ray. Abra o arquivo `zappa_settings.json` e verifique se ele é similar ao exemplo.

```
{
  "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. Adicione "xray_tracing": true como uma entrada ao arquivo de configuração.

```
{
  "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. Implante o aplicativo . Isso configura automaticamente o endpoint do API Gateway e carrega o código no Lambda.

```
zappa deploy
```

```
...
Deploying API Gateway..
Deployment complete!: https://*****.execute-api.us-west-2.amazonaws.com/dev
```

Etapa 3: Habilitar o rastreamento do X-Ray para o API Gateway

Nesta etapa, você interagirá com o console do API Gateway para habilitar o rastreamento do X-Ray.

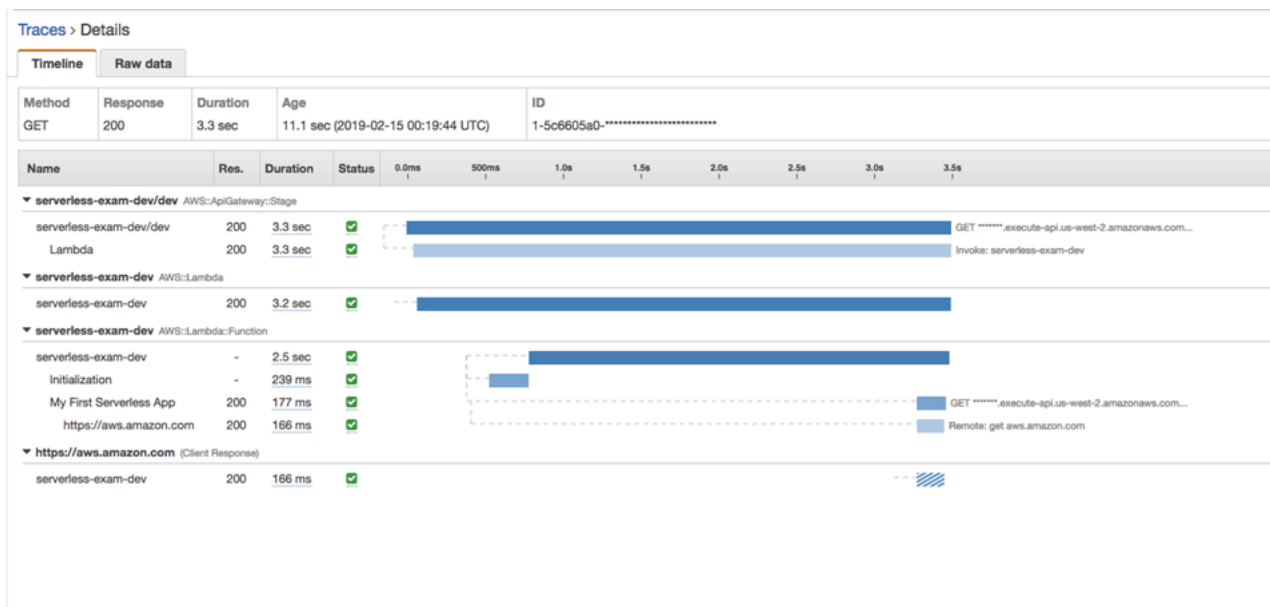
1. Faça login no Console de gerenciamento da AWS e abra o console do API Gateway em <https://console.aws.amazon.com/apigateway/>.
2. Encontre sua API recém-gerada. Ele deve ser semelhante a `serverless-exam-dev`.
3. Escolha Estágios.
4. Escolha o nome de seu estágio de implantação. O padrão é `dev`.
5. Na guia Logs/Tracing (Logs/rastreamento), marque a caixa Enable X-Ray Tracing (Habilitar rastreamento do X-Ray).
6. Escolha Salvar alterações.
7. Acesse o endpoint em seu navegador. Se você usou o aplicativo de exemplo Hello World, ele deve exibir o seguinte.

```
"Hello, World: https://aws.amazon.com/"
```

Etapa 4: Visualizar o rastreamento criado

Nesta etapa, você interagirá com o console do X-Ray para visualizar o rastreamento criado pela aplicação de exemplo. Para obter uma demonstração detalhada sobre a análise de rastreamento, consulte [Visualizar o mapa de serviço](#).

1. Faça login no Console de gerenciamento da AWS e abra o console X-Ray em <https://console.aws.amazon.com/xray/casa>.
2. Visualize os segmentos gerados pelo API Gateway, pela função do Lambda e pelo contêiner do Lambda.
3. No segmento da função do Lambda, visualize um subsegmento chamado `My First Serverless App`. Ele é seguido por um segundo subsegmento chamado `https://aws.amazon.com`.
4. Durante a inicialização, o Lambda também pode gerar um terceiro subsegmento chamado `initialization`.





Etapa 5: limpar

Sempre encerre os recursos que não estão mais em uso para evitar o acúmulo de custos inesperados. Como este tutorial demonstra, ferramentas como o Zappa simplificam a reimplantação sem servidor.

Para remover sua aplicação do Lambda, do API Gateway e do Amazon S3, execute o comando a seguir no diretório do projeto usando a AWS CLI.

```
zappa undeploy dev
```

Próximas etapas

Adicione mais recursos ao seu aplicativo adicionando AWS clientes e instrumentando-os com o X-Ray. Saiba mais sobre as opções de computação sem servidor em [Sem servidor na AWS](#).

Como trabalhar com .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Há duas maneiras de instrumentar um aplicativo .NET para enviar rastreamentos ao X-Ray:

- [AWS Distro for OpenTelemetry .NET](#) — Uma AWS distribuição que fornece um conjunto de bibliotecas de código aberto para enviar métricas e rastreamentos correlacionados para várias soluções de AWS monitoramento, incluindo Amazon e Amazon OpenSearch Service CloudWatch AWS X-Ray, por meio do [AWS Distro](#) for Collector. OpenTelemetry
- [AWS X-Ray SDK para .NET](#) — Um conjunto de bibliotecas para gerar e enviar rastreamentos para o X-Ray por meio do daemon [X-Ray](#).

Para obter mais informações, consulte [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#).

AWS Distro para OpenTelemetry .NET

Com o AWS Distro para OpenTelemetry .NET, você pode instrumentar aplicações uma vez e enviar métricas e rastreamentos correlacionados para várias soluções de monitoramento da AWS, como o Amazon CloudWatch, o AWS X-Ray e o Amazon OpenSearch Service. O uso do X-Ray com o AWS Distro para OpenTelemetry requer dois componentes: um SDK do OpenTelemetry habilitado para uso com o X-Ray e o coletor AWS Distro para OpenTelemetry habilitado para uso com o X-Ray.

Para começar, consulte a documentação do [AWS Distro para OpenTelemetry .NET](#).

Para obter mais informações sobre como usar o AWS Distro para OpenTelemetry com o AWS X-Ray e outros Serviços da AWS, consulte [AWS Distro para OpenTelemetry](#) ou a [documentação do AWS Distro para OpenTelemetry](#).

Para obter mais informações sobre compatibilidade de idiomas e uso, consulte [AWS Observability no GitHub](#).

AWS X-Ray SDK para .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O X-Ray SDK for .NET é uma biblioteca para instrumentar aplicativos web C# .NET, aplicativos web.NET Core e funções do.NET Core em AWS Lambda. Ele fornece classes e métodos para gerar e enviar dados de rastreamento para o [daemon do X-Ray](#). Isso inclui informações sobre solicitações recebidas pelo aplicativo e chamadas que o aplicativo faz para bancos de dados downstream Serviços da AWS, HTTP APIs, web e SQL.

Note

O X-Ray SDK para .NET é um projeto de código aberto. Você pode acompanhar o projeto e enviar problemas e pull requests em GitHub: github.com/aws/aws-xray-sdk-dotnet

Para aplicativos web, comece [adicionando um manipulador de mensagens à sua configuração da web](#) para rastrear solicitações de entrada. O manipulador de mensagens cria um [segmento](#) para cada solicitação rastreada e conclui o segmento quando a resposta é enviada. Embora o segmento esteja aberto, você pode usar os métodos do cliente do SDK para adicionar informações ao segmento e criar subsegmentos para rastrear as chamadas subsequentes. O SDK também registra automaticamente exceções que seu aplicativo lança enquanto o segmento está aberto.

Para funções do Lambda chamadas por um aplicativo ou um serviço instrumentado, o Lambda lê o [cabeçalho de rastreamento](#) e rastreia automaticamente as solicitações amostradas. Para outras funções, você pode [configurar o Lambda](#) para amostrar e rastrear solicitações recebidas. Em ambos os casos, o Lambda cria o segmento e o fornece ao X-Ray SDK.

 Note

No Lambda, o X-Ray SDK é opcional. Se você não o usar em sua função, mesmo assim o mapa de serviço incluirá um nó para o serviço Lambda e um para cada função do Lambda. Ao adicionar o SDK, você pode instrumentar o código da função para adicionar subsegmentos ao segmento de função registrado pelo Lambda. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Depois, use o X-Ray SDK para .NET para [instrumentar clientes do AWS SDK para .NET](#). Sempre que você faz uma chamada para um downstream AWS service (Serviço da AWS) ou recurso com um cliente instrumentado, o SDK registra as informações sobre a chamada em um subsegmento. AWS os serviços e os recursos que você acessa nos serviços aparecem como nós downstream no mapa de rastreamento para ajudá-lo a identificar erros e problemas de limitação em conexões individuais.

[O X-Ray SDK para .NET também fornece instrumentação para chamadas downstream para bancos de dados HTTP APIs Web e SQL](#). O método de extensão `GetResponseTraced` para `System.Net.HttpWebRequest` rastreia chamadas HTTP de saída. Você pode usar a versão `SqlCommand` do X-Ray SDK para .NET para instrumentar consultas SQL.

Depois que você começar a usar o SDK, personalize seu comportamento [configurando o gravador e o manipulador de mensagens](#). Você pode adicionar plug-ins para registrar dados sobre os recursos de computação que executam seu aplicativo, personalizar o comportamento de amostragem, estipulando regras de amostragem, e definir o nível de log para ver mais ou menos informações do SDK nos logs do aplicativo.

Registre informações adicionais sobre as solicitações e o trabalho que o aplicativo faz em [anotações e metadados](#). Anotações são simples pares de chave-valor que são indexados para serem usados com [expressões de filtro](#) para que você possa pesquisar rastreamentos que contêm dados específicos. As entradas de metadados são menos restritivas e podem registrar matrizes e objetos inteiros: tudo o que pode ser serializado em JSON.

Anotações e metadados

Anotações e metadados são textos arbitrários que você adiciona aos segmentos com o X-Ray SDK. As anotações são indexadas para serem usadas com expressões de filtro. Os metadados não são indexados, mas podem ser visualizados no segmento bruto com o console ou a API do X-Ray. Qualquer pessoa à qual você conceder acesso de leitura ao X-Ray poderá visualizar esses dados.

Quando você tem uma grande quantidade de clientes instrumentados no seu código, um único segmento de solicitação pode conter um grande número de subsegmentos, um para cada chamada feita com um cliente instrumentado. Você pode organizar e agrupar subsegmentos integrando chamadas de clientes em [subsegmentos personalizados](#). Você pode criar um subsegmento personalizado para uma função inteira ou qualquer seção de código e registrar metadados e anotações no subsegmento em vez de gravar tudo no segmento principal.

Para documentação de referência sobre as classes e métodos do SDK, consulte o seguinte:

- [AWS X-Ray Referência da API SDK for .NET](#)
- [AWS X-Ray Referência da API principal do SDK for .NET](#)

O mesmo pacote oferece suporte a .NET e .NET Core, mas as classes usadas variam. Os exemplos neste capítulo estão vinculados à referência da API do .NET, a menos que a classe seja específica para o .NET Core.

Requisitos

O X-Ray SDK para .NET requer o .NET Framework 4.5 ou posterior e o AWS SDK para .NET

Para aplicativos e funções do .NET Core, o SDK requer o .NET Core 2.0 ou uma versão posterior.

Adicionar o X-Ray SDK para .NET ao aplicativo

Use NuGet para adicionar o X-Ray SDK para .NET ao seu aplicativo.

Para instalar o X-Ray SDK para .NET NuGet com o gerenciador de pacotes no Visual Studio

1. Escolha Tools, NuGet Package Manager, Manage NuGet Packages for Solution.

2. Pesquise o AWSXRaygravador.
3. Escolha o pacote e, em seguida, escolha Install.

Gerenciamento de dependências

O X-Ray SDK para .NET está disponível no [NuGet](#). Instale o SDK usando o gerenciador de pacotes:

```
Install-Package AWSXRayRecorder -Version 2.10.1
```

O pacote AWSXRayRecorder v2.10.1 do NuGet tem as seguintes dependências:

NET Framework 4.5

```
AWSXRayRecorder (2.10.1)
|
|-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- AWSSDK.Core (>= 3.3.25.1)
|   |
|   |-- AWSXRayRecorder.Handlers.AspNet (>= 2.7.3)
|       |-- AWSXRayRecorder.Core (>= 2.10.1)
|       |
|       |-- AWSXRayRecorder.Handlers.AwsSdk (>= 2.8.3)
|           |-- AWSXRayRecorder.Core (>= 2.10.1)
|           |
|           |-- AWSXRayRecorder.Handlers.EntityFramework (>= 1.1.1)
|               |-- AWSXRayRecorder.Core (>= 2.10.1)
|               |-- EntityFramework (>= 6.2.0)
|               |
|               |-- AWSXRayRecorder.Handlers.SqlServer (>= 2.7.3)
|                   |-- AWSXRayRecorder.Core (>= 2.10.1)
|                   |
|                   |-- AWSXRayRecorder.Handlers.System.Net (>= 2.7.3)
|                       |-- AWSXRayRecorder.Core (>= 2.10.1)
```

NET Framework 2.0

```
AWSXRayRecorder (2.10.1)
```

```
|
|-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- AWSSDK.Core (>= 3.3.25.1)
|   |-- Microsoft.AspNetCore.Http (>= 2.0.0)
|   |-- Microsoft.Extensions.Configuration (>= 2.0.0)
|   |-- System.Net.Http (>= 4.3.4)
|
|-- AWSXRayRecorder.Handlers.AspNetCore (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- Microsoft.AspNetCore.Http.Extensions (>= 2.0.0)
|   |-- Microsoft.AspNetCore.Mvc.Abstractions (>= 2.0.0)
|
|-- AWSXRayRecorder.Handlers.AwsSdk (>= 2.8.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.EntityFramework (>= 1.1.1)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- Microsoft.EntityFrameworkCore.Relational (>= 3.1.0)
|
|-- AWSXRayRecorder.Handlers.SqlServer (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- System.Data.SqlClient (>= 4.4.0)
|
|-- AWSXRayRecorder.Handlers.System.Net (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
```

Para obter mais detalhes sobre o gerenciamento de dependências, consulte a documentação da Microsoft sobre a [dependência do NuGet](#) e a [resolução de dependências do NuGet](#).

Configurar o X-Ray SDK para .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Você pode configurar o X-Ray SDK para .NET com plug-ins para incluir informações sobre o serviço em que o aplicativo é executada, modificar o comportamento de amostragem padrão ou adicionar regras de amostragem que se aplicam a solicitações para caminhos específicos.

Para aplicativos web .NET, adicione chaves à seção `appSettings` do seu arquivo `Web.config`.

Example `Web.config`

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Para o .NET Core, crie um arquivo chamado `appsettings.json` com chave de nível superior chamada `XRay`.

Example .NET `appsettings.json`

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin",
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

Em seguida, no código do aplicativo, crie um objeto de configuração e use-o para inicializar o gravador do X-Ray. Faça isso antes de [inicializar o gravador](#).

Example .NET Core `Program.cs`: configuração do gravador

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Se você estiver instrumentando um aplicativo web .NET Core, você também pode passar o objeto de configuração para o método `UseXRay` ao configurar [o manipulador de mensagens](#). Para funções do Lambda, use o método `InitializeInstance` como mostrado acima.

Para obter mais informações sobre a API de configuração do .NET Core, consulte [Configure an ASP.NET Core App](#) no site `docs.microsoft.com`.

Seções

- [Plugins](#)
- [Regras de amostragem](#)
- [Registro \(.NET\)](#)
- [Registro \(.NET Core\)](#)
- [Variáveis de ambiente](#)

Plugins

Use plugins para adicionar dados sobre o serviço que hospeda seu aplicativo.

Plugins

- Amazon EC2 — EC2Plugin adiciona o ID da instância, a zona de disponibilidade e o grupo de CloudWatch registros.
- Elastic Beanstalk: o ElasticBeanstalkPlugin adiciona o nome do ambiente, o rótulo da versão e o ID de implantação.
- Amazon ECS: o ECSPPlugin adiciona o ID do contêiner.

Para usar um plugin, configure o cliente do X-Ray SDK para .NET adicionando a configuração AWSXRayPlugins. Se vários plugins se aplicarem ao seu aplicativo, especifique todos eles na mesma configuração, separados por vírgulas.

Example Web.config – plugins

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin,ElasticBeanstalkPlugin"/>
  </appSettings>
</configuration>
```

Example .NET Core appsettings.json: plugins

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin,ElasticBeanstalkPlugin"
  }
}
```

```
}
```

Regras de amostragem

O SDK usa as regras de amostragem que você define no console do X-Ray para determinar quais solicitações serão registradas. A regra padrão rastreia a primeira solicitação a cada segundo e 5% de todas as solicitações adicionais em todos os serviços que enviam rastreamentos ao X-Ray. [Crie regras adicionais no console do X-Ray](#) para personalizar a quantidade de dados registrados para cada uma dos aplicativos.

O SDK aplica regras personalizadas na ordem em que elas estão definidas. Se uma solicitação corresponder a várias regras personalizadas, o SDK aplicará somente a primeira regra.

Note

Se o SDK não conseguir acessar o X-Ray para obter regras de amostragem, ele reverterá para uma regra local padrão da primeira solicitação a cada segundo e 5% de todas as solicitações adicionais por host. Isso pode ocorrer se o host não tiver permissão para chamar a amostragem APIs ou não conseguir se conectar ao daemon X-Ray, que atua como um proxy TCP para chamadas de API feitas pelo SDK.

Você também pode configurar o SDK para carregar regras de amostragem de um documento JSON. O SDK pode usar regras locais como backup para casos em que a amostragem do X-Ray não está disponível ou usar exclusivamente regras locais.

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  
}
```

Este exemplo define uma regra personalizada e uma regra padrão. A regra personalizada aplica uma taxa de amostragem de 5% sem um número mínimo de solicitações para rastrear os caminhos em `/api/move/`. A regra padrão rastreia a primeira solicitação a cada segundo e 10% das solicitações adicionais.

A desvantagem de definir regras localmente é que o destino fixo é aplicado por instância do gravador de forma independente, em vez de ser gerenciado pelo serviço X-Ray. À medida que você implanta mais hosts, a taxa fixa é multiplicada, dificultando o controle da quantidade de dados registrados.

Ativado AWS Lambda, você não pode modificar a taxa de amostragem. Se sua função for chamada por um serviço instrumentado, as chamadas que geraram solicitações que foram amostradas por esse serviço serão registradas pelo Lambda. Se o rastreamento ativo estiver habilitado e nenhum cabeçalho de rastreamento estiver presente, o Lambda tomará a decisão de amostragem.

Para configurar regras de backup, faça com que o X-Ray SDK para .NET carregue as regras de amostragem de um arquivo com a configuração `SamplingRuleManifest`.

Example .NET Web.config – regras de amostragem

```
<configuration>  
  <appSettings>  
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>  
  </appSettings>  
</configuration>
```

Example .NET Core appsettings.json: regras de amostragem

```
{  
  "XRay": {  
    "SamplingRuleManifest": "sampling-rules.json"  
  }  
}
```

Para usar apenas regras locais, crie o gravador com uma `LocalizedSamplingStrategy`. Se você tiver regras de backup configuradas, remova essa configuração.

Example .NET global.asax: regras de amostragem locais

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
    LocalizedSamplingStrategy("samplingrules.json")).Build();
AWSXRayRecorder.InitializeInstance(recorder: recorder);
```

Example .NET Core Program.cs: regras de amostragem locais

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
    LocalizedSamplingStrategy("sampling-rules.json")).Build();
AWSXRayRecorder.InitializeInstance(configuration, recorder);
```

Registro (.NET)

O X-Ray SDK para .NET usa o mesmo mecanismo de registro do [AWS SDK para .NET](#). Se você já configurou seu aplicativo para registrar a AWS SDK para .NET saída, a mesma configuração se aplica à saída do X-Ray SDK for .NET.

Para configurar o registro, adicione uma seção de configuração chamada `aws` no arquivo `App.config` ou no arquivo `Web.config`.

Example Web.config – registro

```
...
<configuration>
  <configSections>
    <section name="aws" type="Amazon.AWSSection, AWSSDK.Core"/>
  </configSections>
  <aws>
    <logging logTo="Log4Net"/>
  </aws>
</configuration>
```

Para obter mais informações, consulte [Configuração do seu AWS SDK para .NET aplicativo](#) no AWS SDK para .NET Guia do desenvolvedor.

Registro (.NET Core)

O X-Ray SDK para .NET usa as mesmas opções de registro em log do [AWS SDK para .NET](#). Para configurar o registro em log de aplicativos .NET Core, passe a opção de registro em log para o método `AWSXRayRecorder.RegisterLogger`.

Por exemplo, para usar o log4net, crie um arquivo de configuração que define o registrador, o formato de saída e o local do arquivo.

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>
```

Em seguida, crie o registrador e aplique a configuração no código do seu programa.

Example .NET Core Program.cs: registro em log

```
using log4net;
using Amazon.XRay.Recorder.Core;

class Program
{
  private static ILog log;
  static Program()
  {
    var logRepository = LogManager.GetRepository(Assembly.GetEntryAssembly());
    XmlConfigurator.Configure(logRepository, new FileInfo("log4net.config"));
    log = LogManager.GetLogger(typeof(Program));
    AWSXRayRecorder.RegisterLogger(LoggingOptions.Log4Net);
  }
  static void Main(string[] args)
  {
    ...
  }
}
```

Para obter mais informações sobre como configurar o log4net, consulte [Configuration](#) em logging.apache.org.

Variáveis de ambiente

Você pode usar variáveis de ambiente para configurar o X-Ray SDK para .NET. O SDK é compatível com as variáveis a seguir.

- `AWS_XRAY_TRACING_NAME`: defina um nome de serviço para o SDK usar para segmentos. Sobreposição o nome do serviço que você definiu na [estratégia de nomeação de segmentos](#) do filtro do servlet.
- `AWS_XRAY_DAEMON_ADDRESS`: defina o host e a porta do receptor do daemon do X-Ray. Por padrão, o SDK usa `127.0.0.1:2000` para dados de rastreamento (UDP) e para amostragem (TCP). Use essa variável se você tiver configurado o daemon para [escutar em uma porta diferente](#) ou se ele estiver sendo executado em um host diferente.

Formato

- Mesma porta: *address:port*
- Portas diferentes: `tcp:address:port` `udp:address:port`
- `AWS_XRAY_CONTEXT_MISSING`: defina como `RUNTIME_ERROR` para lançar exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.

Valores válidos

- `RUNTIME_ERROR`: lance uma exceção de tempo de execução.
- `LOG_ERROR`: registre um erro e continue (padrão).
- `IGNORE_ERROR`: ignore o erro e continue.

Erros relativos a segmentos ou subsegmentos ausentes poderão ocorrer quando você tentar usar um cliente instrumentado no código de inicialização que é executado quando nenhuma solicitação estiver aberta ou em um código que gere um novo thread.

Instrumentar as solicitações HTTP de entrada com o X-Ray SDK para .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-

Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode usar o X-Ray SDK para rastrear solicitações HTTP recebidas que seu aplicativo atende em uma EC2 instância na Amazon EC2 ou no Amazon AWS Elastic Beanstalk ECS.

Use um manipulador de mensagens para instrumentar solicitações HTTP de entrada. Quando você adiciona o manipulador de mensagens ao aplicativo, o X-Ray SDK para .NET cria um segmento para cada solicitação amostrada. Este segmento inclui tempo, método e disposição da solicitação HTTP. Instrumentação adicional cria subsegmentos sobre esse segmento.

Note

Para AWS Lambda funções, o Lambda cria um segmento para cada solicitação de amostra. Consulte [AWS Lambda and AWS X-Ray](#) para obter mais informações.

Cada segmento tem um nome que identifica o aplicativo no mapa de serviços. O segmento pode ser nomeado estaticamente ou você pode configurar o SDK para nomeá-lo dinamicamente com base no cabeçalho do host na solicitação de entrada. A nomeação dinâmica permite agrupar rastreamentos com base no nome de domínio na solicitação e aplicar um nome padrão se o nome não corresponder ao padrão esperado (por exemplo, se o cabeçalho do host for falsificado).

Solicitações encaminhadas

Se um balanceador de carga ou outro intermediário encaminhar uma solicitação para o aplicativo, o X-Ray obterá o IP do cliente do cabeçalho X-Forwarded-For na solicitação em vez do IP de origem no pacote IP. O IP do cliente registrado para uma solicitação encaminhada pode ser falsificado; portanto, não é digno de confiança.

O manipulador de mensagens cria um segmento para cada solicitação recebida com um bloco http que contém as seguintes informações:

- Método HTTP: GET, POST, PUT, DELETE etc.

- Endereço do cliente: o endereço IP do cliente que enviou a solicitação.
- Código de resposta: o código de resposta HTTP da solicitação concluída.
- Horário: a hora de início (quando a solicitação foi recebida) e a hora de término (quando a resposta foi enviada).
- Agente do usuário: o user-agent da solicitação.
- Tamanho do conteúdo: o content-length da resposta.

Seções

- [Instrumentar solicitações de entrada \(.NET\)](#)
- [Instrumentar solicitações de entrada \(.NET Core\)](#)
- [Configurar uma estratégia de nomeação de segmentos](#)

Instrumentar solicitações de entrada (.NET)

Para solicitar os instrumentos atendidos pelo seu aplicativo, chame `RegisterXRay` no método `Init` do seu arquivo `global.asax`.

Example `global.asax` – manipulador de mensagens

```
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");
        }
    }
}
```

Instrumentar solicitações de entrada (.NET Core)

Para instrumentar as solicitações atendidas pelo aplicativo, chame o método `UseXRay` antes de qualquer outro middleware no método `Configure` de sua classe `Startup`. Preferencialmente,

o middleware do X-Ray deve ser o primeiro middleware a processar a solicitação e o último middleware a processar a resposta no pipeline.

Note

Para o .NET Core 2.0, se você tiver um método `UseExceptionHandler` no aplicativo, chame `UseXRay` após o método `UseExceptionHandler` para garantir que as exceções sejam registradas.

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
    ...
}
```

O método `UseXRay` também pode levar um [objeto de configuração](#) como segundo argumento.

```
app.UseXRay("MyApp", configuration);
```

Configurar uma estratégia de nomeação de segmentos

AWS X-Ray usa um nome de serviço para identificar seu aplicativo e diferenciá-lo dos outros aplicativos, bancos de dados APIs, AWS recursos externos e que seu aplicativo usa. Quando o X-Ray SDK gera segmentos para solicitações recebidas, ele registra o nome do serviço do aplicativo no [campo de nome](#) do segmento.

O X-Ray SDK pode nomear segmentos com o nome do host no cabeçalho da solicitação HTTP. No entanto, esse cabeçalho pode ser falsificado, o que pode resultar em nós inesperados no mapa de serviço. Para evitar que o SDK nomeie segmentos incorretamente devido a solicitações com cabeçalhos de host falsificados, você deve especificar um nome padrão para as solicitações recebidas.

Se o aplicativo atende a solicitações para vários domínios, você pode configurar o SDK para usar uma estratégia de nomeação dinâmica para refletir isso nos nomes dos segmentos. Uma estratégia de nomeação dinâmica permite que o SDK use o nome do host para solicitações que correspondam a um padrão esperado e aplique o nome padrão às solicitações que não correspondem.

Por exemplo, você pode ter um único aplicativo para atender a solicitações para três subdomínios: `www.example.com`, `api.example.com` e `static.example.com`. Você pode usar uma estratégia de nomeação dinâmica com o padrão `*.example.com` a fim de identificar segmentos para cada subdomínio com um nome diferente, o que resulta em três nós de serviço no mapa de serviços. Se o aplicativo receber solicitações com um nome de host que não corresponda ao padrão, você verá um quarto nó no mapa de serviços com um nome alternativo especificado por você.

Para usar o mesmo nome para todos os segmentos de solicitação, especifique o nome do seu aplicativo ao inicializar o manipulador de mensagens, como mostrado na [seção anterior](#). Isso tem o mesmo efeito que criar um [FixedSegmentNamingStrategy](#) e passá-lo para o método `RegisterXRay`.

```
AWSXRayAspNet.RegisterXRay(this, new FixedSegmentNamingStrategy("MyApp"));
```

Note

Você pode sobrepor o nome do serviço padrão que definiu no código com a [variável de ambiente](#) `AWS_XRAY_TRACING_NAME`.

Uma estratégia de nomenclatura dinâmica define um padrão que os nomes de host devem corresponder, e um nome padrão a ser usado se o nome do host na solicitação HTTP não corresponder ao padrão. Para nomear segmentos dinamicamente, crie um [DynamicSegmentNamingStrategy](#) e pass-o para o método `RegisterXRay`.

```
AWSXRayAspNet.RegisterXRay(this, new DynamicSegmentNamingStrategy("MyApp",  
    "*.example.com"));
```

Rastreando chamadas AWS do SDK com o X-Ray SDK for .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

[Quando seu aplicativo faz chamadas Serviços da AWS para armazenar dados, gravar em uma fila ou enviar notificações, o X-Ray SDK para .NET rastreia as chamadas downstream em subsegmentos.](#)

Os recursos rastreados Serviços da AWS e que você acessa nesses serviços (por exemplo, um bucket do Amazon S3 ou uma fila do Amazon SQS) aparecem como nós downstream no mapa de rastreamento no console X-Ray.

Você pode instrumentar todos os seus AWS SDK para .NET clientes ligando `RegisterXRayForAllServices` antes de criá-los.

Example SampleController.cs - instrumentação do cliente DynamoDB

```
using Amazon;  
using Amazon.Util;  
using Amazon.DynamoDBv2;  
using Amazon.DynamoDBv2.DocumentModel;  
using Amazon.XRay.Recorder.Core;  
using Amazon.XRay.Recorder.Handlers.AwsSdk;
```

```
namespace SampleEBWebApplication.Controllers
{
    public class SampleController : ApiController
    {
        AWS SDKHandler.RegisterXRayForAllServices();
        private static readonly Lazy<AmazonDynamoDBClient> LazyDdbClient = new
        Lazy<AmazonDynamoDBClient>(() =>
        {
            var client = new AmazonDynamoDBClient(EC2InstanceMetadata.Region ??
            RegionEndpoint.USEast1);
            return client;
        });
    }
}
```

Para instrumentar clientes para alguns serviços, mas não outros, chame `RegisterXRay` em vez de `RegisterXRayForAllServices`. Substitua o texto destacado pelo nome da interface do cliente do serviço.

```
AWS SDKHandler.RegisterXRay<IAmazonDynamoDB>()
```

Para todos os serviços, o nome da API chamada no console do X-Ray pode ser visto. Para um subconjunto de serviços, o X-Ray SDK adiciona informações ao segmento para fornecer maior detalhamento no mapa de serviços.

Por exemplo, quando você faz uma chamada com um cliente instrumentado do DynamoDB, o SDK adiciona o nome da tabela ao segmento para chamadas direcionadas a uma tabela. No console, cada tabela aparece como um nó separado no mapa de serviços, com um nó genérico do DynamoDB para chamadas não direcionadas a uma tabela.

Example Subsegmento para uma chamada ao DynamoDB para salvar um item

```
{
  "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": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",  
}  
}
```

Ao acessar recursos nomeados, as chamadas para os serviços os seguir criam nós adicionais no mapa de serviço. As chamadas que não apontam para recursos específicos criam um nó genérico para o serviço.

- Amazon DynamoDB: nome da tabela
- Amazon Simple Storage Service: nome de chave e bucket
- Amazon Simple Queue Service: nome da fila

Rastrear chamadas para serviços web HTTP subsequentes com o X-Ray SDK para .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando seu aplicativo faz chamadas para microsserviços ou HTTP público APIs, você pode usar o método de `GetResponseTraced` extensão X-Ray SDK for .NET `System.Net.HttpWebRequest` para instrumentar essas chamadas e adicionar a API ao gráfico de serviços como um serviço downstream.

Example `HttpWebRequest`

```
using System.Net;
```

```
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.System.Net;

private void MakeHttpRequest()
{
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://names.example.com/
api");
    request.GetResponseTraced();
}
```

Para chamadas assíncronas, use `GetAsyncResponseTraced`.

```
request.GetAsyncResponseTraced();
```

Se você usa o [system.net.http.httpclient](https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient), use o manipulador de delegação `HttpClientXRayTracingHandler` para registrar as chamadas.

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);
}
```

Quando você instrumenta uma chamada para uma API subsequente da web, o X-Ray SDK para .NET registra um subsegmento com informações sobre a solicitação e a resposta HTTP. O X-Ray usa o subsegmento para gerar um segmento inferido para a API.

Example Subsegmento para uma chamada HTTP downstream

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
```

```
"http": {
  "request": {
    "method": "GET",
    "url": "https://names.example.com/"
  },
  "response": {
    "content_length": -1,
    "status": 200
  }
}
```

Example Segmento inferido para uma chamada HTTP de downstream

```
{
  "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
}
```

Rastrear consultas SQL com o X-Ray SDK para .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray](#)

[SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

O SDK fornece uma classe wrapper para `System.Data.SqlClient.SqlCommand`, denominada `TraceableSqlCommand`, que você pode usar em vez de `SqlCommand`. É possível inicializar um comando SQL com a classe `TraceableSqlCommand`.

Rastrear consultas SQL com métodos síncronos e assíncronos

Os exemplos a seguir mostram como usar o `TraceableSqlCommand` para rastrear automaticamente consultas do SQL Server de forma síncrona e assíncrona.

Example **Controller.cs** – Instrumentação de cliente SQL (síncrono)

```
using Amazon;
using Amazon.Util;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.SqlServer;

private void QuerySql(int id)
{
    var connectionString = ConfigurationManager.AppSettings["RDS_CONNECTION_STRING"];
    using (var sqlConnection = new SqlConnection(connectionString))
        using (var sqlCommand = new TraceableSqlCommand("SELECT " + id, sqlConnection))
        {
            sqlCommand.Connection.Open();
            sqlCommand.ExecuteNonQuery();
        }
}
```

É possível executar a consulta de forma assíncrona usando o método `ExecuteReaderAsync`.

Example **Controller.cs** – instrumentação de cliente SQL (assíncrono)

```
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();
}
}
```

Coletar consultas SQL feitas ao SQL Server

É possível habilitar a captura de `SqlCommand.CommandText` como parte do subsegmento criado pela consulta SQL. O `SqlCommand.CommandText` aparece como o campo `sanitized_query` no subsegmento JSON. Por padrão, esse recurso está desabilitado para segurança.

Note

Não habilite o recurso de coleta se você estiver incluindo informações confidenciais como texto simples em suas consultas SQL.

É possível habilitar a coleção de consultas SQL de duas maneiras:

- Defina a propriedade `CollectSqlQueries` como `true` na configuração global do aplicativo.
- Defina o parâmetro `collectSqlQueries` na instância `TraceableSqlCommand` como `true` para coletar chamadas dentro da instância.

Ativar a `CollectSqlQueries` propriedade global

Os exemplos a seguir mostram como habilitar a propriedade `CollectSqlQueries` para .NET e .NET Core.

.NET

Para definir a propriedade `CollectSqlQueries` como `true` na configuração global do seu aplicativo em .NET, modifique o appsettings do arquivo `Web.config` ou `App.config`, conforme mostrado.

Example **App.config** ou **Web.config**: habilitar a coleta de consultas SQL globalmente

```
<configuration>
```

```
<appSettings>
  <add key="CollectSqlQueries" value="true">
</appSettings>
</configuration>
```

.NET Core

Para definir a propriedade `CollectSqlQueries` como `true` na configuração global do aplicativo no .NET Core, modifique o arquivo `appsettings.json` na chave do X-Ray, conforme mostrado.

Example **appsettings.json**: habilitar a coleta de consultas SQL globalmente

```
{
  "XRay": {
    "CollectSqlQueries": "true"
  }
}
```

Ative o `collectSqlQueries` parâmetro

É possível definir o parâmetro `collectSqlQueries` na instância `TraceableSqlCommand` como `true` para coletar o texto da consulta SQL para consultas do SQL Server feitas usando essa instância. Definir o parâmetro como `false` desabilita o recurso `CollectSqlQuery` para a instância `TraceableSqlCommand`.

Note

O valor de `collectSqlQueries` na instância `TraceableSqlCommand` substitui o valor definido na configuração global da propriedade `CollectSqlQueries`.

Example Exemplo de **Controller.cs**: habilitar a coleta de consultas SQL para a instância

```
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();
    }
}
```

Criar subsegmentos adicionais

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Os subsegmentos estendem o [segmento](#) de um rastreamento com detalhes sobre o trabalho realizado para atender a uma solicitação. Sempre que você faz uma chamada com um cliente instrumentado, o X-Ray SDK registra as informações geradas em um subsegmento. Você pode criar subsegmentos adicionais para agrupar outros subsegmentos, medir o desempenho de uma seção do código ou registrar anotações e metadados.

Para gerenciar subsegmentos, use os métodos `BeginSubsegment` e `EndSubsegment`. Execute qualquer trabalho no subsegmento em um bloco de `try` e use `AddException` para rastrear exceções. Chame `EndSubsegment` em um bloco `finally` para garantir que o subsegmento seja fechado.

Example Controller.cs: subsegmento personalizado

```
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
try
{
```

```
    DoWork();
}
catch (Exception e)
{
    AWSXRayRecorder.Instance.AddException(e);
}
finally
{
    AWSXRayRecorder.Instance.EndSubsegment();
}
```

Quando você cria um subsegmento dentro de um segmento ou outro subsegmento, o X-Ray SDK para .NET gera um ID para ele e registra a hora de início e de término.

Example Subsegmento com metadados

```
"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"
    }
  },
}
```

Adicionar anotações e metadados aos segmentos com o X-Ray SDK para .NET

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Você pode registrar informações adicionais sobre solicitações, o ambiente ou o aplicativo com anotações e metadados. Também é possível adicionar anotações e metadados aos segmentos que o X-Ray SDK cria ou aos subsegmentos personalizados que você cria.

Anotações são pares de chave-valor com valores booleanos, de string ou número. As anotações são indexadas para serem usadas com [expressões de filtro](#). Use anotações para registrar dados que você deseja usar para agrupar rastreamentos no console ou ao chamar a API [GetTraceSummaries](#).

Metadados são pares chave-valor que podem ter valores de qualquer tipo, incluindo objetos e listas, mas não são indexados para uso com expressões de filtro. Use metadados para registrar dados adicionais que você deseja armazenar no rastreamento e não precisa usar com a pesquisa.

Seções

- [Registrar anotações com o X-Ray SDK para .NET](#)
- [Registrar metadados com o X-Ray SDK para .NET](#)

Registrar anotações com o X-Ray SDK para .NET

Use anotações para registrar informações em segmentos ou subsegmentos que você deseja indexar para pesquisa.

O seguinte é necessário para todas as anotações no X-Ray:

Requisitos de anotação

- Chaves: a chave para uma anotação do X-Ray pode ter até 500 caracteres alfanuméricos. Você não pode usar espaços ou símbolos que não sejam um ponto (.)
- Valores: o valor de uma anotação do X-Ray pode ter até 1.000 caracteres Unicode.
- O número de anotações: você pode usar até cinquenta anotações por rastreamento.

Para gravar anotações fora de uma função AWS Lambda

1. Obtenha uma instância do `AWSXRayRecorder`.

```
using Amazon.XRay.Recorder.Core;  
...  
AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
```

2. Chame `addAnnotation` com uma chave de string e um booleano, `Int32`, `Int64`, `Double` ou valor de string.

```
recorder.AddAnnotation("mykey", "my value");
```

O exemplo a seguir mostra como chamar `putAnnotation` com uma chave `String` que inclui um ponto e um valor booleano, numérico ou de string.

```
document.putAnnotation("testkey.test", "my value");
```

Para gravar anotações dentro de uma função AWS Lambda

Tanto os segmentos quanto os subsegmentos dentro de uma função do Lambda são gerenciados pelo ambiente de runtime do Lambda. Para adicionar uma anotação a um segmento ou subsegmento dentro de uma função do Lambda, faça o seguinte:

1. Crie o segmento ou subsegmento dentro da função do Lambda.
2. Adicione a anotação ao segmento ou subsegmento.
3. Finalize o segmento ou subsegmento.

O exemplo de código a seguir mostra como adicionar uma anotação a um subsegmento dentro de uma função do Lambda:

```
#Create the subsegment
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
#Add an annotation
AWSXRayRecorder.Instance.AddAnnotation("My", "Annotation");
try
{
    YourProcess(); #Your function
}
catch (Exception e)
{
    AWSXRayRecorder.Instance.AddException(e);
}
finally #End the subsegment
{
    AWSXRayRecorder.Instance.EndSubsegment();
}
```

```
}
```

O X-Ray SDK registra anotações como pares de chave-valor em um objeto `annotations` no documento de segmentos. Chamar a operação `addAnnotation` duas vezes com a mesma chave substitui um valor registrado anteriormente no mesmo segmento ou subsegmento.

Para encontrar rastreamentos que têm anotações com valores específicos, use a palavra-chave `annotation[key]` em uma [expressão de filtro](#).

Registrar metadados com o X-Ray SDK para .NET

Use metadados para registrar informações em segmentos ou subsegmentos os quais você não precisa indexar para uso em uma pesquisa. Valores de metadados podem ser strings, números, booleanos ou outro objeto que possa ser serializado em uma matriz ou objeto JSON.

Como registrar metadados

1. Obtenha uma instância de `AWSXRayRecorder`, conforme mostrado no seguinte exemplo de código:

```
using Amazon.XRay.Recorder.Core;  
...  
AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
```

2. Chame `AddMetadata` com um namespace de string, uma chave de string e um valor de objeto, conforme mostrado no exemplo de código a seguir:

```
recorder.AddMetadata("my namespace", "my key", "my value");
```

Você também pode chamar a operação `AddMetadata` usando apenas um par de chave e valor, conforme mostrado no seguinte exemplo de código:

```
recorder.AddMetadata("my key", "my value");
```

Se você não especificar um valor para o namespace, o X-Ray SDK usará `default`. Chamar a operação `AddMetadata` duas vezes com a mesma chave substitui um valor registrado anteriormente no mesmo segmento ou subsegmento.

Trabalhar com Ruby

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Há duas maneiras de instrumentar um aplicativo Ruby para enviar rastreamentos ao X-Ray:

- [AWS Distro for OpenTelemetry Ruby](#) — Uma AWS distribuição que fornece um conjunto de bibliotecas de código aberto para enviar métricas e rastreamentos correlacionados para várias soluções de AWS monitoramento, incluindo Amazon AWS X-Ray e Amazon OpenSearch Service CloudWatch, por meio do [AWS Distro](#) for Collector. OpenTelemetry
- [AWS X-Ray SDK for Ruby](#) — Um conjunto de bibliotecas para gerar e enviar rastreamentos para o X-Ray por meio do daemon X-Ray.

Para obter mais informações, consulte [Escolhendo entre o AWS Distro for OpenTelemetry e o X-Ray SDKs](#).

AWS Distro para OpenTelemetry Ruby

Com o AWS Distro para OpenTelemetry (ADOT) Ruby, você pode instrumentar as aplicações uma vez e enviar métricas e rastreamentos correlacionados a várias soluções de monitoramento, como o Amazon CloudWatch, o AWS X-Ray e o Amazon OpenSearch Service. O uso do X-Ray com o ADOT requer dois componentes: um SDK do OpenTelemetry habilitado para uso com o X-Ray e o coletor AWS Distro para OpenTelemetry habilitado para uso com o X-Ray.

Para começar, consulte a [documentação do AWS Distro para OpenTelemetry Ruby](#).

Para obter mais informações sobre como usar o AWS Distro para OpenTelemetry com o AWS X-Ray e outros Serviços da AWS, consulte [AWS Distro para OpenTelemetry](#) ou a [documentação do AWS Distro para OpenTelemetry](#).

Para obter mais informações sobre compatibilidade de idiomas e uso, consulte [AWSobservability no GitHub](#).

AWS X-Ray SDK para Ruby

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

O X-Ray SDK é uma biblioteca para aplicações web do Ruby que fornece classes e métodos para gerar e enviar dados de rastreamento ao daemon do X-Ray. Os dados de rastreamento incluem informações sobre solicitações HTTP recebidas pelo aplicativo e chamadas que o aplicativo faz para serviços downstream usando o AWS SDK, clientes HTTP ou um cliente de registro ativo. Você também pode criar segmentos manualmente e adicionar informações de depuração em anotações e metadados.

Você pode baixar do SDK, adicionando-o ao seu gemfile e executando `bundle install`.

Example Gemfile

```
gem 'aws-sdk'
```

Se você usa o Rails, primeiro [adicione o middleware do X-Ray SDK](#) para rastrear solicitações de entrada. Um filtro de solicitação cria um [segmento](#). Embora o segmento esteja aberto, você pode usar os métodos do cliente do SDK para adicionar informações ao segmento e criar subsegmentos para rastrear as chamadas subsequentes. O SDK também registra automaticamente exceções que

seu aplicativo lança enquanto o segmento está aberto. Para aplicativos que não sejam Rails, você pode [criar segmentos manualmente](#).

Em seguida, use o X-Ray SDK para instrumentar seus AWS SDK para Ruby clientes HTTP e SQL [configurando o gravador para corrigir as](#) bibliotecas associadas. Sempre que você faz uma chamada para um downstream AWS service (Serviço da AWS) ou recurso com um cliente instrumentado, o SDK registra as informações sobre a chamada em um subsegmento. Serviços da AWS e os recursos que você acessa nos serviços aparecem como nós downstream no mapa de rastreamento para ajudá-lo a identificar erros e problemas de limitação em conexões individuais.

Assim que você começar a usar o SDK, personalize seu comportamento [configurando o gravador](#). Você pode adicionar plug-ins para registrar dados sobre os recursos de computação que executam seu aplicativo, personalizar o comportamento de amostragem definindo as regras de amostragem e fornecer um registrador para ver mais ou menos informações do SDK nos logs do seu aplicativo.

Registre informações adicionais sobre as solicitações e o trabalho que o a aplicação faz em [anotações e metadados](#). Anotações são simples pares de chave-valor que são indexados para serem usados com [expressões de filtro](#) para que você possa pesquisar rastreamentos que contêm dados específicos. As entradas de metadados são menos restritivas e podem registrar matrizes e objetos inteiros: tudo o que pode ser serializado em JSON.

Anotações e metadados

Anotações e metadados são textos arbitrários que você adiciona aos segmentos com o X-Ray SDK. As anotações são indexadas para serem usadas com expressões de filtro. Os metadados não são indexados, mas podem ser visualizados no segmento bruto com o console ou a API do X-Ray. Qualquer pessoa à qual você conceder acesso de leitura ao X-Ray poderá visualizar esses dados.

Quando há uma grande quantidade de clientes instrumentados no código, um único segmento de solicitação pode conter um grande número de subsegmentos, um para cada chamada feita com um cliente instrumentado. Você pode organizar e agrupar subsegmentos integrando chamadas de clientes em [subsegmentos personalizados](#). Você pode criar um subsegmento personalizado para uma função inteira ou qualquer seção de código e registrar metadados e anotações no subsegmento em vez de gravar tudo no segmento principal.

Para documentação de referência para as classes e métodos do SDK, consulte a [Referência de API do AWS X-Ray X-Ray SDK para Ruby](#).

Requisitos

O X-Ray SDK requer o Ruby 2.3 ou posterior e é compatível com as seguintes bibliotecas:

- AWS SDK para Ruby versão 3.0 ou posterior
- Rails versão 5.1 ou posterior

Configurar o X-Ray SDK para Ruby

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

O X-Ray SDK para Ruby tem uma classe chamada `XRayRecorder`, que fornece o gravador global. Você pode configurar o gravador global para personalizar o middleware que cria segmentos para chamadas HTTP de entrada.

Seções

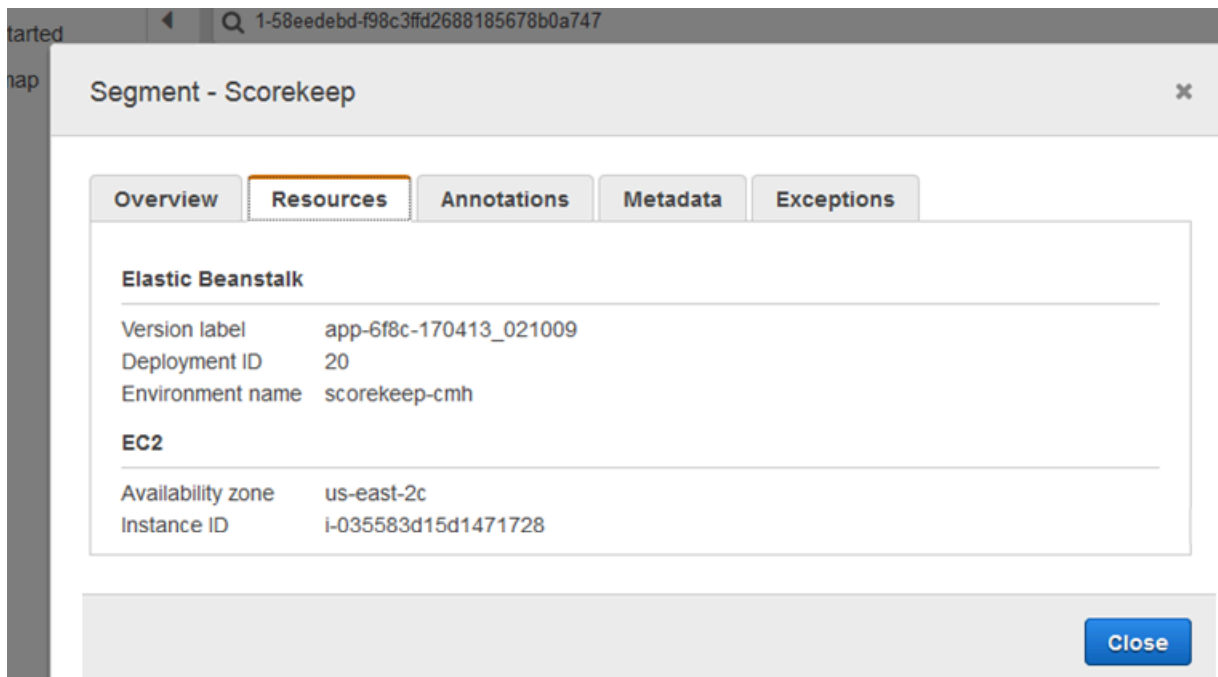
- [Plug-ins de serviço](#)
- [Regras de amostragem](#)
- [Registro em log](#)
- [Configuração do gravador no código](#)
- [Configuração do gravador com o rails](#)
- [Variáveis de ambiente](#)

Plug-ins de serviço

Use plugins para registrar informações sobre o serviço que hospeda o aplicativo.

Plugins

- Amazon EC2 — `ec2` adiciona o ID da instância e a zona de disponibilidade.
- Elastic Beanstalk: o `elastic_beanstalk` adiciona o nome do ambiente, o rótulo da versão e o ID de implantação.
- Amazon ECS: o `ecs` adiciona o ID do contêiner.



Para usar plug-ins, você deve especificá-los no objeto de configuração que passa ao gravador.

Example main.rb: configuração do plug-in

```
my_plugins = %I[ec2 elastic_beanstalk]

config = {
  plugins: my_plugins,
  name: 'my app',
}

XRay.recorder.configure(config)
```

Você também pode usar [variáveis de ambiente](#), que têm precedência sobre valores definidos no código, para configurar o gravador.

O SDK também usa as configurações do plug-in para definir o campo `origin` no segmento. Isso indica o tipo de AWS recurso que executa seu aplicativo. Quando você usa vários plug-ins, o SDK usa a seguinte ordem de resolução para determinar a origem: ElasticBeanstalk > EKS > ECS > EC2

Regras de amostragem

O SDK usa as regras de amostragem que você define no console do X-Ray para determinar quais solicitações serão registradas. A regra padrão rastreia a primeira solicitação a cada segundo e 5% de todas as solicitações adicionais em todos os serviços que enviam rastreamentos ao X-Ray. [Crie regras adicionais no console do X-Ray](#) para personalizar a quantidade de dados registrados para cada uma dos aplicativos.

O SDK aplica regras personalizadas na ordem em que elas estão definidas. Se uma solicitação corresponder a várias regras personalizadas, o SDK aplicará somente a primeira regra.

Note

Se o SDK não conseguir acessar o X-Ray para obter regras de amostragem, ele reverterá para uma regra local padrão da primeira solicitação a cada segundo e 5% de todas as solicitações adicionais por host. Isso pode ocorrer se o host não tiver permissão para chamar a amostragem APIs ou não conseguir se conectar ao daemon X-Ray, que atua como um proxy TCP para chamadas de API feitas pelo SDK.

Você também pode configurar o SDK para carregar regras de amostragem de um documento JSON. O SDK pode usar regras locais como backup para casos em que a amostragem do X-Ray não está disponível ou usar exclusivamente regras locais.

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  
  }  
}
```

Este exemplo define uma regra personalizada e uma regra padrão. A regra personalizada aplica uma taxa de amostragem de 5% sem um número mínimo de solicitações para rastrear os caminhos em `/api/move/`. A regra padrão rastreia a primeira solicitação a cada segundo e 10% das solicitações adicionais.

A desvantagem de definir regras localmente é que o destino fixo é aplicado por instância do gravador de forma independente, em vez de ser gerenciado pelo serviço X-Ray. À medida que você implanta mais hosts, a taxa fixa é multiplicada, dificultando o controle da quantidade de dados registrados.

Para configurar regras de backup, defina um hash para o documento no objeto de configuração que você passa para o gravador.

Example main.rb: configuração da regra de backup

```
require 'aws-xray-sdk'  
my_sampling_rules = {  
  version: 1,  
  default: {  
    fixed_target: 1,  
    rate: 0.1  
  }  
}  
config = {  
  sampling_rules: my_sampling_rules,  
  name: 'my app',  
}  
XRay.recorder.configure(config)
```

Para armazenar as regras de amostragem de forma independente, defina o hash em um arquivo separado e exija que o arquivo o inclua em seu aplicativo.

Example config/sampling-rules.rb

```
my_sampling_rules = {
```

```
version: 1,  
default: {  
  fixed_target: 1,  
  rate: 0.1  
}  
}
```

Example main.rb: regra de amostragem de um arquivo

```
require 'aws-xray-sdk'  
require 'config/sampling-rules.rb'  
  
config = {  
  sampling_rules: my_sampling_rules,  
  name: 'my app',  
}  
XRay.recorder.configure(config)
```

Para usar apenas regras locais, exija as regras de amostragem e configure o `LocalSampler`.

Example main.rb: amostragem de regra local

```
require 'aws-xray-sdk'  
require 'aws-xray-sdk/sampling/local/sampler'  
  
config = {  
  sampler: LocalSampler.new,  
  name: 'my app',  
}  
XRay.recorder.configure(config)
```

Você também pode configurar o gravador global para desabilitar a amostragem e instrumentar todas as solicitações de entrada.

Example main.rb: desabilitar a amostragem

```
require 'aws-xray-sdk'  
config = {  
  sampling: false,  
  name: 'my app',  
}  
XRay.recorder.configure(config)
```

Registro em log

Por padrão, o gravador encaminha os eventos informativos para `$stdout`. Você pode personalizar o registro em log definindo um [registrador](#) no objeto de configuração que você passa para o gravador.

Example main.rb: registrar em log

```
require 'aws-xray-sdk'
config = {
  logger: my_logger,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Use logs de depuração para identificar problemas como subsegmentos não fechados ao [gerar subsegmentos manualmente](#).

Configuração do gravador no código

Configurações adicionais estão disponíveis no método `configure` no `XRay.recorder`.

- `context_missing`: defina como `LOG_ERROR` para evitar o lançamento de exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.
- `daemon_address`: defina o host e a porta do receptor do daemon do X-Ray.
- `name`: defina um nome de serviço para o SDK usar para segmentos.
- `naming_pattern`: defina um nome de domínio padrão para usar a [nomeação dinâmica](#).
- `plugins`: registre informações sobre os recursos da AWS de sua aplicação com [plug-ins](#).
- `sampling`: defina como `false` para desabilitar a amostragem.
- `sampling_rules`: defina o hash que contém suas [regras de amostragem](#).

Example main.rb: desabilitar exceções de contexto ausente

```
require 'aws-xray-sdk'
config = {
  context_missing: 'LOG_ERROR'
}

XRay.recorder.configure(config)
```


Configuração do gravador com o rails

Se você usa o framework do Rails, pode configurar as opções do gravador global em um arquivo Ruby em `app_root/initializers`. O X-Ray SDK aceita uma chave de configuração adicional para usar com o Rails.

- `active_record`: defina como `true` para registrar os subsegmentos das transações de banco de dados de registros ativos.

Defina as configurações disponíveis em um objeto de configuração denominado `Rails.application.config.xray`.

Example `config/initializers/aws_xray.rb`

```
Rails.application.config.xray = {  
  name: 'my app',  
  patch: %I[net_http aws_sdk],  
  active_record: true  
}
```

Variáveis de ambiente

É possível usar variáveis de ambiente para configurar o X-Ray SDK para Ruby. O SDK é compatível com as seguintes variáveis:

- `AWS_XRAY_TRACING_NAME`: defina um nome de serviço para o SDK usar para segmentos. Sobrepõe o nome do serviço que você definiu na [estratégia de nomeação de segmentos](#) do filtro do servlet.
- `AWS_XRAY_DAEMON_ADDRESS`: defina o host e a porta do receptor do daemon do X-Ray. Por padrão, o SDK envia dados de rastreamento para `127.0.0.1:2000`. Use essa variável se você tiver configurado o daemon para [escutar em uma porta diferente](#) ou se ele estiver sendo executado em um host diferente.
- `AWS_XRAY_CONTEXT_MISSING`: defina como `RUNTIME_ERROR` para lançar exceções, caso o código instrumentado tente registrar dados quando nenhum segmento estiver aberto.

Valores válidos

- `RUNTIME_ERROR`: lance uma exceção de tempo de execução.
- `LOG_ERROR`: registre um erro e continue (padrão).

- `IGNORE_ERROR`: ignore o erro e continue.

Erros relativos a segmentos ou subsegmentos ausentes poderão ocorrer quando você tentar usar um cliente instrumentado no código de inicialização que é executado quando nenhuma solicitação estiver aberta ou em um código que gere um novo thread.

As variáveis de ambiente substituem os valores definidos no código.

Rastrear solicitações de entrada com o middleware do X-Ray SDK para Ruby

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode usar o X-Ray SDK para rastrear solicitações HTTP recebidas que seu aplicativo atende em uma EC2 instância na Amazon EC2 ou no Amazon AWS Elastic Beanstalk ECS.

Se você usa o Rails, use o middleware do Rails para instrumentar as solicitações HTTP de entrada. Quando você adiciona o middleware ao aplicativo e configura o nome do segmento, o X-Ray SDK para Ruby cria um segmento para cada solicitação amostrada. Todos os segmentos criados por instrumentação adicional tornam-se subsegmentos do segmento em nível da solicitação que fornecem informações sobre solicitação e resposta HTTP. Essas informações incluem tempo, método e disposição da solicitação.

Cada segmento tem um nome que identifica a aplicação no mapa de serviços. O segmento pode ser nomeado estaticamente ou você pode configurar o SDK para nomeá-lo dinamicamente com base no cabeçalho do host na solicitação de entrada. A nomeação dinâmica permite agrupar rastreamentos com base no nome de domínio na solicitação e aplicar um nome padrão se o nome não corresponder ao padrão esperado (por exemplo, se o cabeçalho do host for falsificado).

Solicitações encaminhadas

Se um balanceador de carga ou outro intermediário encaminhar uma solicitação para o aplicativo, o X-Ray obterá o IP do cliente do cabeçalho `X-Forwarded-For` na solicitação em vez do IP de origem no pacote IP. O IP do cliente registrado para uma solicitação encaminhada pode ser falsificado; portanto, não é digno de confiança.

Quando uma solicitação é encaminhada, o SDK define um campo adicional no segmento para indicar isso. Se o segmento tiver o campo `x_forwarded_for` definido como `true`, isso significa que o IP do cliente foi obtido no cabeçalho `X-Forwarded-For` na solicitação HTTP.

O middleware cria um segmento para cada solicitação de entrada com um bloco `http` contendo as seguintes informações:

- Método HTTP: GET, POST, PUT, DELETE etc.
- Endereço do cliente: o endereço IP do cliente que enviou a solicitação.
- Código de resposta: o código de resposta HTTP da solicitação concluída.
- Horário: a hora de início (quando a solicitação foi recebida) e a hora de término (quando a resposta foi enviada).
- Agente do usuário: o `user-agent` da solicitação.
- Tamanho do conteúdo: o `content-length` da resposta.

Como usar o middleware do Rails

Para usar o middleware, atualize o `gemfile` para incluir o [railtie](#) necessário.

Example Gemfile: rails

```
gem 'aws-xray-sdk', require: ['aws-xray-sdk/facets/rails/railtie']
```

Para usar o middleware, você também deve [configurar o gravador](#) com um nome que representa a aplicação no mapa de rastreamento.

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {
```

```
name: 'my app'
}
```

Instrumentação de código manual

Se você não usa o Rails, crie os segmentos manualmente. Você pode criar um segmento para cada solicitação recebida ou criar segmentos em torno de clientes HTTP ou AWS SDK corrigidos para fornecer contexto para que o gravador adicione subsegmentos.

```
# Start a segment
segment = XRay.recorder.begin_segment 'my_service'
# Start a subsegment
subsegment = XRay.recorder.begin_subsegment 'outbound_call', namespace: 'remote'

# Add metadata or annotation here if necessary
my_annotations = {
  k1: 'v1',
  k2: 1024
}
segment.annotations.update my_annotations

# Add metadata to default namespace
subsegment.metadata[:k1] = 'v1'

# Set user for the segment (subsegment is not supported)
segment.user = 'my_name'

# End segment/subsegment
XRay.recorder.end_subsegment
XRay.recorder.end_segment
```

Configurar uma estratégia de nomeação de segmentos

AWS X-Ray usa um nome de serviço para identificar seu aplicativo e diferenciá-lo dos outros aplicativos, bancos de dados APIs, AWS recursos externos e que seu aplicativo usa. Quando o X-Ray SDK gera segmentos para solicitações recebidas, ele registra o nome do serviço do aplicativo no [campo de nome](#) do segmento.

O X-Ray SDK pode nomear segmentos com o nome do host no cabeçalho da solicitação HTTP. No entanto, esse cabeçalho pode ser falsificado, o que pode resultar em nós inesperados no mapa de serviço. Para evitar que o SDK nomeie segmentos incorretamente devido a solicitações

com cabeçalhos de host falsificados, você deve especificar um nome padrão para as solicitações recebidas.

Se o aplicativo atende a solicitações para vários domínios, você pode configurar o SDK para usar uma estratégia de nomeação dinâmica para refletir isso nos nomes dos segmentos. Uma estratégia de nomeação dinâmica permite que o SDK use o nome do host para solicitações que correspondam a um padrão esperado e aplique o nome padrão às solicitações que não correspondem.

Por exemplo, você pode ter um único aplicativo para atender a solicitações para três subdomínios: `www.example.com`, `api.example.com` e `static.example.com`. Você pode usar uma estratégia de nomeação dinâmica com o padrão `*.example.com` a fim de identificar segmentos para cada subdomínio com um nome diferente, o que resulta em três nós de serviço no mapa de serviços. Se a aplicação receber solicitações com um nome de host que não corresponda ao padrão, você verá um quarto nó no mapa de serviços com um nome alternativo especificado por você.

Para usar o mesmo nome para todos os segmentos de solicitação, especifique o nome da aplicação ao configurar o gravador, conforme mostrado nas [seções anteriores](#).

Uma estratégia de nomeação dinâmica define um padrão com o qual os nomes de host devem corresponder e um nome padrão a ser usado se o nome do host na solicitação HTTP não corresponder ao padrão. Para nomear segmentos dinamicamente, especifique um padrão de nomeação no hash do config.

Example `main.rb`: nomeação dinâmica

```
config = {  
  naming_pattern: '*mydomain*',  
  name: 'my app',  
}
```

```
XRay.recorder.configure(config)
```

Você pode usar `*` no padrão para fazer a correspondência com qualquer string ou `?` para um caractere único.

Note

Você pode sobrepor o nome do serviço padrão que definiu no código com a [variável de ambiente](#) `AWS_XRAY_TRACING_NAME`.

Aplicar patches a bibliotecas para instrumentar chamadas downstream

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Para instrumentar chamadas subsequentes, use o X-Ray SDK para Ruby para aplicar patches às bibliotecas usadas pela aplicação. O X-Ray SDK para Ruby pode corrigir as bibliotecas a seguir.

Bibliotecas com suporte

- [net/http](#): instrumentar clientes HTTP.
- [aws-sdk](#)— AWS SDK para Ruby Clientes de instrumentos.

Quando você usa uma biblioteca com patches aplicados, o X-Ray SDK para Ruby cria um subsegmento para a chamada e registra as informações da solicitação e da resposta. O segmento deve estar disponível para que o SDK crie o subsegmento do middleware do SDK ou de uma chamada para `XRay.recorder.begin_segment`.

Quanto às bibliotecas de patches, é necessário especificá-las no objeto de configuração que você passa ao gravador do X-Ray.

Example main.rb: bibliotecas de patches

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http aws_sdk]
}

XRay.recorder.configure(config)
```

Rastreando chamadas AWS do SDK com o X-Ray SDK for Ruby

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Quando seu aplicativo faz chamadas Serviços da AWS para armazenar dados, gravar em uma fila ou enviar notificações, o X-Ray SDK for Ruby rastreia as chamadas downstream em subsegmentos.

Os recursos rastreados Serviços da AWS e que você acessa nesses serviços (por exemplo, um bucket do Amazon S3 ou uma fila do Amazon SQS) aparecem como nós downstream no mapa de rastreamento no console X-Ray.

O X-Ray SDK for Ruby AWS instrumenta automaticamente todos os clientes SDK [quando você](#) corrige a biblioteca. aws - sdk Você não pode instrumentar clientes individuais.

Para todos os serviços, o nome da API chamada no console do X-Ray pode ser visto. Para um subconjunto de serviços, o X-Ray SDK adiciona informações ao segmento para fornecer maior detalhamento no mapa de serviços.

Por exemplo, quando você faz uma chamada com um cliente instrumentado do DynamoDB, o SDK adiciona o nome da tabela ao segmento para chamadas direcionadas a uma tabela. No console, cada tabela aparece como um nó separado no mapa de serviços, com um nó genérico do DynamoDB para chamadas não direcionadas a uma tabela.

Example Subsegmento para uma chamada ao DynamoDB para salvar um item

```
{
  "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": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Ao acessar recursos nomeados, as chamadas para os serviços os seguir criam nós adicionais no mapa de serviço. As chamadas que não apontam para recursos específicos criam um nó genérico para o serviço.

- Amazon DynamoDB: nome da tabela
- Amazon Simple Storage Service: nome de chave e bucket
- Amazon Simple Queue Service: nome da fila

Gerar subsegmentos personalizados com o X-Ray SDK

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o OpenTelemetry. Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Os subsegmentos estendem o [segmento](#) de um rastreamento com detalhes sobre o trabalho realizado para atender a uma solicitação. Sempre que você faz uma chamada com um cliente instrumentado, o X-Ray SDK registra as informações geradas em um subsegmento. Você pode criar subsegmentos adicionais para agrupar outros subsegmentos, medir o desempenho de uma seção do código ou registrar anotações e metadados.

Para gerenciar subsegmentos, use os métodos `begin_subsegment` e `end_subsegment`.

```
subsegment = XRay.recorder.begin_subsegment name: 'annotations', namespace: 'remote'
my_annotations = { id: 12345 }
subsegment.annotations.update my_annotations
XRay.recorder.end_subsegment
```

Para criar um subsegmento para uma função, encapsule-o em uma chamada para `XRay.recorder.capture`.

```
XRay.recorder.capture('name_for_subsegment') do |subsegment|
  resp = myfunc() # myfunc is your function
  subsegment.annotations.update k1: 'v1'
  resp
end
```

Quando você cria um subsegmento dentro de um segmento ou outro subsegmento, o X-Ray SDK gera um ID para ele e registra a hora de início e de término.

Exemplo Subsegmento com metadados

```
"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"
    }
  },
}
```

Adicionar anotações e metadados aos segmentos com o X-Ray SDK para Ruby

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais

informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#). Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Você pode registrar informações adicionais sobre solicitações, o ambiente ou o aplicativo com anotações e metadados. Também é possível adicionar anotações e metadados aos segmentos que o X-Ray SDK cria ou aos subsegmentos personalizados que você cria.

Anotações são pares de chave-valor com valores booleanos, de string ou número. As anotações são indexadas para serem usadas com [expressões de filtro](#). Use anotações para registrar dados que você deseja usar para agrupar rastreamentos no console ou ao chamar a API [GetTraceSummaries](#).

Metadados são pares chave-valor que podem ter valores de qualquer tipo, incluindo objetos e listas, mas não são indexados para uso com expressões de filtro. Use metadados para registrar dados adicionais que você deseja armazenar no rastreamento e não precisa usar com a pesquisa.

Além de anotações e metadados, você também pode [registrar strings de ID de usuário](#) em segmentos. IDs Os usuários são registrados em um campo separado nos segmentos e são indexados para uso com a pesquisa.

Seções

- [Registrar anotações com o X-Ray SDK para Ruby](#)
- [Registrar metadados com o X-Ray SDK para Ruby](#)
- [Gravando usuário IDs com o X-Ray SDK for Ruby](#)

Registrar anotações com o X-Ray SDK para Ruby

Use anotações para registrar informações em segmentos ou subsegmentos que você deseja indexar para pesquisa.

Requisitos de anotação

- Chaves: a chave para uma anotação do X-Ray pode ter até 500 caracteres alfanuméricos. Você não pode usar espaços ou símbolos que não sejam um ponto (.)
- Valores: o valor de uma anotação do X-Ray pode ter até 1.000 caracteres Unicode.

- O número de anotações: você pode usar até cinquenta anotações por rastreamento.

Como registrar anotações

1. Obtenha uma referência para o segmento ou subsegmento atual no `xray_recorder`.

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

ou

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_subsegment
```

2. Chame `update` com um valor de hash.

```
my_annotations = { id: 12345 }
document.annotations.update my_annotations
```

Veja a seguir um exemplo que mostra como chamar `update` com uma chave de anotação que contém um ponto.

```
my_annotations = { testkey.test: 12345 }
document.annotations.update my_annotations
```

O SDK registra anotações como pares de chave-valor em um objeto `annotations` no documento de segmentos. Chamar `add_annotations` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Para encontrar rastreamentos que têm anotações com valores específicos, use a palavra-chave `annotation[key]` em uma [expressão de filtro](#).

Registrar metadados com o X-Ray SDK para Ruby

Use metadados para registrar informações em segmentos ou subsegmentos dos quais você não precisa indexados para pesquisa. Valores de metadados podem ser strings, números, booleanos ou qualquer objeto que possa ser serializado em uma matriz ou objeto JSON.

Como registrar metadados

1. Obtenha uma referência para o segmento ou subsegmento atual no `xray_recorder`.

```
require 'aws-xray-sdk'  
...  
document = XRay.recorder.current_segment
```

ou

```
require 'aws-xray-sdk'  
...  
document = XRay.recorder.current_subsegment
```

2. Chame `metadata` com uma chave de string, um booleano, um número, uma string ou valor de objeto e um namespace de string.

```
my_metadata = {  
  my_namespace: {  
    key: 'value'  
  }  
}  
subsegment.metadata my_metadata
```

Chamar `metadata` duas vezes com a mesma chave substitui os valores registrados anteriormente no mesmo segmento ou subsegmento.

Gravando usuário IDs com o X-Ray SDK for Ruby

Registre o usuário IDs nos segmentos da solicitação para identificar o usuário que enviou a solicitação.

Para gravar usuário IDs

1. Obtenha uma referência para o segmento atual em `xray_recorder`.

```
require 'aws-xray-sdk'  
...  
document = XRay.recorder.current_segment
```

2. Defina o campo do usuário no segmento com o ID da string do usuário que enviou a solicitação.

```
segment.user = 'U12345'
```

Você pode configurar o usuário em seus controladores para registrar o ID de usuário assim que o aplicativo começar a processar uma solicitação.

Para encontrar rastreamentos para um ID de usuário, use a `user` palavra-chave em uma [expressão de filtragem](#).

Cronograma de suporte do X-Ray SDK e do Daemon Support

A tabela a seguir lista as datas e o nível de suporte para X-Ray SDKs e Daemon.

Fase do SDK e do daemon	Data de início	Data de término	Suporte fornecido
Disponibilidade geral	NA	25 de fevereiro de 2026	X-Ray SDKs e Daemon são totalmente suportados. AWS fornece lançamentos regulares de SDK e daemon que incluem correções de bugs e segurança.
Modo de manutenção	25 de fevereiro de 2026	N/D	AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Eles não SDKs/Daemon receberão novos aprimoramentos de recursos.

Recomendamos que você migre para OpenTelemetry soluções para instrumentar seu aplicativo e enviar rastreamentos para o X-Ray AWS . Para obter mais informações sobre a migração para OpenTelemetry, consulte [Migrando da instrumentação X-Ray para a instrumentação](#). OpenTelemetry

Migração da instrumentação X-Ray para a instrumentação OpenTelemetry

Note

Aviso de SDK/Daemon manutenção do X-Ray — Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará as versões do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre a linha do tempo do suporte, consulte [Cronograma de suporte do X-Ray SDK e do Daemon Support](#).

O X-Ray está migrando para OpenTelemetry (OTel) como seu principal padrão de instrumentação para rastreamento e observabilidade de aplicativos. Essa mudança estratégica se alinha AWS às melhores práticas do setor e oferece aos clientes uma solução mais abrangente, flexível e pronta para o futuro para suas necessidades de observabilidade. OpenTelemetryA ampla adoção da X-Ray no setor permite o rastreamento de solicitações em diversos sistemas, incluindo aqueles externos AWS que podem não se integrar diretamente ao X-Ray.

Este capítulo fornece recomendações para uma transição suave e enfatiza a importância de migrar para soluções OpenTelemetry baseadas para garantir suporte e acesso contínuos aos recursos mais recentes em instrumentação e observabilidade de aplicativos.

É recomendável adotar OpenTelemetry como solução de observabilidade para instrumentar seu aplicativo.

Tópicos

- [Compreensão OpenTelemetry](#)
- [Entendendo OpenTelemetry os conceitos de migração](#)
- [Visão geral da migração](#)
- [Migração do X-Ray Daemon para AWS CloudWatch agente ou coletor OpenTelemetry](#)
- [Migrando para Java OpenTelemetry](#)
- [Migre para Go OpenTelemetry](#)
- [Migrar para Node.js OpenTelemetry](#)
- [Migrar para o.NET OpenTelemetry](#)

- [Migrar para Python OpenTelemetry](#)
- [Migrar para Ruby OpenTelemetry](#)

Compreensão OpenTelemetry

OpenTelemetry é uma estrutura de observabilidade padrão do setor que fornece protocolos e ferramentas padronizados para coletar dados de telemetria. Ele oferece uma abordagem unificada para instrumentar, gerar, coletar e exportar dados de telemetria, como métricas, logs e rastreamentos.

Ao migrar do X-Ray SDKs para OpenTelemetry, você obtém os seguintes benefícios:

- Suporte aprimorado para estrutura e instrumentação de biblioteca
- Suporte para linguagens de programação adicionais
- Recursos de instrumentação automática
- Opções flexíveis de configuração de amostragem
- Coleta unificada de métricas, logs e rastreamentos

O OpenTelemetry coletor fornece mais opções para formatos de coleta de dados e destinos de exportação do que o daemon X-Ray.

OpenTelemetry suporte em AWS

AWS fornece várias soluções para trabalhar com OpenTelemetry:

- AWS Distro para OpenTelemetry

Exporte OpenTelemetry traços como segmentos para o X-Ray.

Para obter mais informações, consulte [AWS Distro for OpenTelemetry](#).

- CloudWatch Sinais de aplicação

Exporte OpenTelemetry traços e métricas personalizados para monitorar a integridade do aplicativo.

Para obter mais informações, consulte [Trabalhar com o Application Signals](#).

- CloudWatch OTel Ponto final

Exporte OpenTelemetry traços para o X-Ray usando o OTel endpoint HTTP com OpenTelemetry instrumentação nativa.

Para obter mais informações, consulte [Usando OTel endpoints](#).

Usando OpenTelemetry com AWS CloudWatch

AWS CloudWatch oferece suporte a OpenTelemetry rastreamentos por meio de instrumentação de aplicativos do lado do cliente e AWS CloudWatch serviços nativos, como Application Signals, Trace, Map, Metrics e Logs. Para obter mais informações, consulte [OpenTelemetry](#).

Entendendo OpenTelemetry os conceitos de migração

A tabela a seguir mapeia os conceitos de X-Ray para seus OpenTelemetry equivalentes. A compreensão desses mapeamentos ajuda você a traduzir sua instrumentação de X-Ray existente para: OpenTelemetry

Conceito do X-Ray	OpenTelemetry conceito
Gravador do X-Ray	Tracer Provider e rastreadores
Plug-ins de serviço	Detector de recursos
Segmento	Extensão (servidor)
Subsegmento	Extensão (sem servidor)
Regras de amostragem do X-Ray	OpenTelemetry Amostragem (personalizável)
Emissor do X-Ray	Exportador de extensão (personalizável)
Anotações/metadados	Atributos
Instrumentação da biblioteca	Instrumentação da biblioteca
Contexto de rastreamento do X-Ray	Contexto da extensão
Propagação do contexto de rastreamento do X-Ray	Propagação do contexto de rastreamento do W3C

Conceito do X-Ray	OpenTelemetry conceito
Amostragem de rastreamento do X-Ray Trace	OpenTelemetry Amostragem de traços
N/D	Processamento da extensão
N/D	Baggage
Daemon do X-Ray	OpenTelemetry Colecionador

 Note

Para obter mais informações sobre OpenTelemetry conceitos, consulte a [OpenTelemetry documentação](#).

Comparando recursos

As tabelas a seguir mostram quais recursos são compatíveis com os dois serviços. Use essas informações para identificar lacunas que você precisará resolver durante a migração:

Recurso	Instrumentação do X-Ray	OpenTelemetry instrumen tação
Instrumentação da biblioteca	Compatível	Compatível
Amostragem do X-Ray	Compatível	Compatível com OTel java/.net/GO Compatível com ADOT Java/. NET/Python/Node.js
Propagação do contexto de rastreamento do X-Ray	Compatível	Compatível
Deteccção de recursos	Compatível	Compatível
Anotações de segmento	Compatível	Compatível

Recurso	Instrumentação do X-Ray	OpenTelemetry instrumentação
Metadados de segmento	Compatível	Compatível
Instrumentação automática de código zero	Compatível com Java	Compatível com OTel Java/.NET/Python/Node.js Compatível com ADOT Java/.NET/Python/Node.js
Criação manual de rastreamento	Compatível	Compatível

Configurar o rastreamento

Para criar rastreamentos OpenTelemetry, você precisa de um rastreador. Para obter um rastreador, inicialize o Trace Provider na sua aplicação. Isso é semelhante à maneira como você usa o gravador do X-Ray para configurar o X-Ray e criar segmentos e subsegmentos em um rastreamento de X-Ray.

Note

O OpenTelemetry Tracer Provider oferece mais opções de configuração do que o X-Ray Recorder.

Noções básicas sobre a estrutura dos dados de rastreamento

Após entender os conceitos básicos e os mapeamentos de recursos, você pode saber mais sobre detalhes específicos de implementação, como estrutura de dados de rastreamento e amostragem.

OpenTelemetry usa extensões em vez de segmentos e subsegmentos para estruturar dados de rastreamento. Cada extensão inclui os seguintes componentes:

- Nome
- ID exclusivo
- Carimbos de data/hora de início e término

- Tipo de extensão
- Contexto da extensão
- Atributos (metadados de valores-chave)
- Eventos (logs de data e hora)
- Links para outras extensões
- Informações de status
- Referências de extensão principal

Quando você migra para OpenTelemetry, suas extensões são automaticamente convertidas em segmentos ou subsegmentos de X-Ray. Isso garante que sua experiência de CloudWatch console existente permaneça inalterada.

Trabalhar com atributos de extensão

O SDK do X-Ray fornece duas maneiras de adicionar dados a segmentos e subsegmentos:

Anotações

Pares de valores-chave que são indexados para filtragem e pesquisa

Metadados

Pares de valores-chave que contêm dados complexos que não são indexados para pesquisa

Por padrão, os atributos OpenTelemetry span são convertidos em metadados nos dados brutos do X-Ray. Para converter atributos específicos em anotações, adicione as chaves deles à lista de atributos `aws.xray.annotations`.

- Para obter mais informações sobre OpenTelemetry conceitos, consulte [OpenTelemetry Traces](#)
- Para obter detalhes sobre como os OpenTelemetry dados são mapeados para dados do X-Ray, consulte [OpenTelemetry Tradução do modelo de dados do X-Ray](#)

Detectar recursos no seu ambiente

OpenTelemetry usa Detectores de recursos para coletar metadados sobre os recursos que geram dados de telemetria. Esses metadados são armazenados como Atributos de recursos. Por exemplo, uma entidade que produz telemetria pode ser um cluster do Amazon ECS ou uma EC2 instância

da Amazon, e os atributos de recursos que podem ser registrados a partir dessas entidades podem incluir o ARN do cluster do Amazon ECS ou o ID da instância da Amazon. EC2

- Para obter informações sobre os tipos de recursos compatíveis, consulte Convenções [semânticas de OpenTelemetry recursos](#)
- Para obter informações sobre os plug-ins de serviço do X-Ray, consulte [Configurar o SDK do X-Ray](#).

Gerenciar estratégias de amostragem

A amostragem de rastreamento ajuda você a gerenciar custos coletando dados de um subconjunto representativo de solicitações, em vez de todas as solicitações. Tanto o X-Ray OpenTelemetry quanto o X-Ray oferecem suporte à amostragem, mas a implementam de forma diferente.

Note

A amostragem de menos de 100% dos rastreamentos reduz seus custos de observabilidade, mantendo insights significativos sobre o desempenho da sua aplicação.

OpenTelemetry fornece várias estratégias de amostragem integradas e permite criar estratégias personalizadas. Você também pode configurar um X-Ray Remote Sampler em algumas linguagens do SDK para usar as regras de amostragem do X-Ray. OpenTelemetry

As estratégias de amostragem adicionais de OpenTelemetry são:

- Amostragem baseada no pai: respeita a decisão de amostragem da extensão pai antes de aplicar outras estratégias de amostragem
- Amostragem baseada na taxa de identificação do rastreamento: faz uma amostragem aleatória de uma porcentagem especificada de extensões
- Amostragem da cauda — Aplica regras de amostragem para completar traços no coletor OpenTelemetry
- Amostradores personalizados: implemente sua própria lógica de amostragem usando a interface de amostragem

Para obter informações sobre as regras de amostragem do X-Ray, consulte [Regras de amostragem no console do X-Ray](#)

Para obter informações sobre amostragem de OpenTelemetry cauda, consulte Processador de amostragem de [cauda](#)

Gerenciar o contexto de rastreamento

O X-Ray SDKs gerencia o contexto do segmento para lidar corretamente com os relacionamentos pai-filho entre segmentos e subsegmentos em um rastreamento. OpenTelemetry usa um mecanismo semelhante para garantir que as extensões tenham a extensão principal correta. Ele armazena e propaga dados de rastreamento em todo o contexto de uma solicitação. Por exemplo, quando seu aplicativo processa uma solicitação e cria uma extensão de servidor para representar essa solicitação, OpenTelemetry armazenará a extensão do servidor no OpenTelemetry Contexto para que, quando uma extensão secundária for criada, essa extensão secundária possa referenciar a extensão no Contexto como sua principal.

Propagar o contexto de rastreamento

Tanto o X-Ray quanto OpenTelemetry o uso de cabeçalhos HTTP para propagar o contexto de rastreamento entre os serviços. Isso permite vincular dados de rastreamento gerados por diferentes serviços e manter as decisões de amostragem.

O SDK do X-Ray propaga automaticamente o contexto de rastreamento usando o cabeçalho de rastreamento do X-Ray. Quando um serviço chama outro, o cabeçalho de rastreamento contém o contexto necessário para manter os relacionamentos pai-filho entre os rastreamentos.

OpenTelemetry suporta vários formatos de cabeçalho de rastreamento para propagação de contexto, incluindo:

- Contexto de rastreamento do W3C (padrão)
- Cabeçalho de rastreamento do X-Ray
- Outros formatos personalizados

Note

Você pode configurar OpenTelemetry para usar um ou mais formatos de cabeçalho. Por exemplo, use o X-Ray Propagator para enviar contexto de rastreamento para AWS serviços que oferecem suporte ao rastreamento de X-Ray.

Configure e use o X-Ray Propagator para permitir o rastreamento entre AWS serviços. Isso permite que você propague o contexto de rastreamento para endpoints do API Gateway e outros serviços compatíveis com o X-Ray.

- Para obter informações sobre cabeçalhos de rastreamento do X-Ray, consulte [Cabeçalho de rastreamento](#) no Guia do desenvolvedor do X-Ray
- Para obter informações sobre propagação de OpenTelemetry contexto, consulte [Contexto e propagação de contexto](#) na documentação OpenTelemetry

Usar instrumentação de biblioteca

Tanto o X-Ray quanto o X-Ray OpenTelemetry fornecem instrumentação de biblioteca que requer alterações mínimas no código para adicionar rastreamento aos seus aplicativos.

O X-Ray fornece funcionalidades de instrumentação de biblioteca. Isso permite que você adicione instrumentações do X-Ray predefinidas com o mínimo de alterações no código da aplicação. Essas instrumentações oferecem suporte a bibliotecas específicas, como AWS SDK e clientes HTTP, bem como estruturas da web como Spring Boot ou Express.js.

OpenTelemetryAs bibliotecas de instrumentação da geram extensões detalhadas para suas bibliotecas por meio de ganchos de biblioteca ou modificação automática de código, exigindo o mínimo de alterações no código.

[Para determinar se o Library Instrumentations OpenTelemetry é compatível com sua biblioteca, procure-o no OpenTelemetry Registro em OpenTelemetry Registro.](#)

Exportar rastreamentos

X-Ray e OpenTelemetry use métodos diferentes para exportar dados de rastreamento.

Exportação de rastreamento do X-Ray

O X-Ray SDKs usa um emissor para enviar dados de rastreamento:

- Envia segmentos e subsegmentos para o daemon do X-Ray
- Usa UDP para E/S sem bloqueio
- É configurado por padrão no SDK

OpenTelemetry exportação de rastreamento

OpenTelemetry usa exportadores de Span configuráveis para enviar dados de rastreamento:

- Usa os protocolos http/protobuf ou grpc
- Exporta extensões para endpoints monitorados pelo OpenTelemetry coletor ou agente CloudWatch
- Permite configurações personalizadas do exportador

Processamento e encaminhamento de rastreamento

Tanto o X-Ray quanto o X-Ray OpenTelemetry fornecem componentes para receber, processar e encaminhar dados de rastreamento.

Processamento de rastreamentos do X-Ray

O daemon do X-Ray gerencia o processamento de rastreamentos:

- Ouve o tráfego UDP do X-Ray SDKs
- Agrupa segmentos e subsegmentos
- Carrega lotes para o serviço do X-Ray

OpenTelemetry processamento de traços

O OpenTelemetry Collector manipula o processamento de traços:

- Recebe rastreamentos de serviços instrumentados
- Processa e, opcionalmente, modifica os dados de rastreamento
- Envia rastreamentos processados para vários backends, incluindo o X-Ray

Note

O AWS CloudWatch agente também pode receber e enviar OpenTelemetry traços para o X-Ray. Para obter mais informações, consulte [Coletar métricas e rastreamentos com OpenTelemetry](#).

Processamento de extensão (conceito OpenTelemetry específico)

OpenTelemetry usa processadores de extensão para modificar extensões à medida que elas são criadas:

- Permite ler e modificar extensões na criação ou na conclusão
- Habilita a lógica personalizada para o gerenciamento de extensões

Bagagem (conceito específico) OpenTelemetry

OpenTelemetryO recurso de bagagem do permite a propagação de dados de valores-chave:

- Permite a transmissão de dados arbitrários com o contexto de rastreamento
- É útil para propagar informações específicas da aplicação para todos os limites do serviço

[Para obter informações sobre o OpenTelemetry Collector, consulte OpenTelemetry Collector](#)

Para obter informações sobre os conceitos do X-Ray, consulte [Conceitos do X-Ray](#) no Guia do desenvolvedor do X-Ray

Visão geral da migração

Esta seção fornece uma visão geral das alterações de código necessárias para a migração. A lista abaixo contém orientações específicas do idioma e as etapas de migração do daemon do X-Ray.

Important

Para migrar totalmente da instrumentação X-Ray para a OpenTelemetry instrumentação, você precisa:

1. Substitua o uso do X-Ray SDK por uma solução OpenTelemetry
2. Substitua o X-Ray Daemon pelo CloudWatch agente ou OpenTelemetry coletor (com o X-Ray Exporter)

- [Migrando para Java OpenTelemetry](#)
- [Migre para Go OpenTelemetry](#)

- [Migrar para Node.js OpenTelemetry](#)
- [Migrar para o.NET OpenTelemetry](#)
- [Migrar para Python OpenTelemetry](#)
- [Migrar para Ruby OpenTelemetry](#)

Recomendações de aplicações novas e existentes

Para aplicações novas e existentes, é recomendável usar as seguintes soluções para permitir o rastreamento nas suas aplicações:

Instrumentação

- OpenTelemetry SDKs
- AWS Distro para instrumentação OpenTelemetry

Coleta de dados

- OpenTelemetry Colecionador
- CloudWatch Agente

Depois de migrar para soluções OpenTelemetry baseadas, sua CloudWatch experiência permanecerá a mesma. Você ainda poderá visualizar seus traços no mesmo formato nas páginas Traces e Trace Map do CloudWatch console ou recuperar seus dados de rastreamento por meio do [X-Ray APIs](#).

Alterações na configuração de rastreamento

Você precisa substituir a configuração do X-Ray por uma OpenTelemetry configuração.

Comparação entre X-Ray e OpenTelemetry configuração

Recurso	SDK do X-Ray	OpenTelemetry
Configurações padrão	• Amostragem centralizada do X-Ray • Propagação do contexto de rastreamento do X-Ray • Exportação de rastreamento para o daemon do X-Ray	• Exportação de rastreamentos para o OpenTelemetry Collector ou CloudWatch Agent (HTTP/gRPC) • Propagação do contexto de rastreamento do W3C

Recurso	SDK do X-Ray	OpenTelemetry
Configurações manuais	• Regras de amostragem locais • Plug-ins de detecção de recursos	• Amostragem do X-Ray (pode não estar disponível para todos os idiomas) • Detecção de recursos • Propagação do contexto de rastreamento do X-Ray

Alterações na instrumentação de biblioteca

Atualize seu código para usar o OpenTelemetry Library Instrumentation em vez do X-Ray Library Instrumentation para AWS SDK, clientes HTTP, Web Frameworks e outras bibliotecas. Isso gera OpenTelemetry Traces em vez de X-Ray Traces.

Note

As alterações no código variam conforme o idioma e a biblioteca. Consulte os guias de migração específicos do idioma para obter instruções detalhadas.

Alterações na instrumentação do ambiente do Lambda

Para usar OpenTelemetry em suas funções do Lambda, escolha uma dessas opções de configuração:

1. Usar uma camada do Lambda de instrumentação automática:

- Camada [Lambda \(recomendada\) de sinais de CloudWatch aplicação](#)

Note

Para usar somente o rastreamento, defina a variável de ambiente `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false` do Lambda.

- [AWS Lambda Layer gerenciada para ADOT](#)

2. Configurado manualmente OpenTelemetry para sua função Lambda:

- Configure um processador de extensão simples com um exportador de extensão UDP do X-Ray
- Configure um propagador do X-Ray para Lambda

Criar dados de rastreamento manualmente

Substitua segmentos e subsegmentos do X-Ray por OpenTelemetry Spans:

- Use um OpenTelemetry rastreador para criar vãos
- Adicione atributos às extensões (equivalentes aos metadados e anotações do X-Ray)

Important

Quando enviado para o X-Ray:

- As extensões do servidor são convertidas em segmentos do X-Ray
- Outras extensões são convertidas em subsegmentos do X-Ray
- Os atributos são convertidos em metadados por padrão

Para converter um atributo em uma anotação, adicione a chave à lista de atributos `aws.xray.annotations`. Para obter mais informações, consulte [Enable The Customized X-Ray Annotations](#).

Migração do X-Ray Daemon para AWS CloudWatch agente ou coletor OpenTelemetry

Você pode usar o CloudWatch agente ou o OpenTelemetry coletor para receber traços de seus aplicativos instrumentados e enviá-los para o X-Ray.

Note

A versão 1.300025.0 e posterior do CloudWatch agente pode coletar rastreamentos. OpenTelemetry Usar o CloudWatch agente em vez do X-Ray Daemon reduz o número de agentes que você precisa gerenciar. Para obter mais informações, consulte [Coleta de métricas, registros e rastreamentos com o CloudWatch agente](#).

Seções

- [Migração na Amazon EC2 ou em servidores locais](#)
- [Migrar no Amazon ECS](#)
- [Migrar no Elastic Beanstalk](#)

Migração na Amazon EC2 ou em servidores locais

Important

Pare o processo do X-Ray Daemon antes de usar o CloudWatch agente ou OpenTelemetry coletor para evitar conflitos de portas.

Configuração existente do daemon do X-Ray

Instalar o daemon

Seu uso atual do daemon do X-Ray foi instalado usando um destes métodos:

Instalação manual

Baixar e executar o arquivo executável do bucket do Amazon S3 do daemon do X-Ray.

Instalação automática

Usar este script para instalar o daemon ao iniciar uma instância:

```
#!/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
```

Configurar o daemon do

Seu uso atual do daemon do X-Ray foi configurado usando:

- Argumentos da linha de comando
- Arquivo de configuração (`xray-daemon.yaml`)

Example Usar um arquivo de configuração

```
./xray -c ~/xray-daemon.yaml
```

Execução do daemon

Seu uso atual do daemon do X-Ray foi iniciado com o seguinte comando:

```
~/xray-daemon$ ./xray -o -n us-east-1
```

Remover o daemon

Para remover o Daemon X-Ray da sua instância da Amazon EC2 :

1. Interrompa o serviço do daemon:

```
systemctl stop xray
```

2. Exclua o arquivo de configuração:

```
rm ~/path/to/xray-daemon.yaml
```

3. Se configurado, remova o arquivo de log:

Note

A localização do arquivo de log depende da sua configuração:

- Configuração da linha de comando: `/var/log/xray-daemon.log`
- Arquivo de configuração: verifique a configuração de `LogPath`

Configurando o CloudWatch agente

Instalar o agente

Para obter instruções de instalação, consulte [Instalação do CloudWatch agente em um servidor local](#).

Configurar o agente

1. Crie um arquivo de configuração para habilitar a coleta de rastreamentos. Para obter mais informações, consulte [Criação do arquivo de configuração do CloudWatch agente](#).

2. Configurar permissões do IAM:

- Anexe um perfil do IAM ou especifique as credenciais do agente. Para obter mais informações, consulte [Configurar perfis do IAM](#).
- Verifique se o perfil ou as credenciais incluem a permissão `xray:PutTraceSegments`.

Como iniciar o agente do

Para obter instruções sobre como iniciar o agente, consulte [Iniciando o CloudWatch agente usando a linha de comando](#).

Configurando o OpenTelemetry coletor

Instalar o coletor

Baixe e instale o OpenTelemetry coletor para seu sistema operacional. Para obter instruções, consulte [Install the Collector](#).

Configurar o coletor

Configure os seguintes componentes em seu coletor:

- Extensão awsproxy

Necessário para amostragem do X-Ray

- OTel receptores

Coleta rastreamentos da sua aplicação

- Exportador xray

Envia rastreamentos para o X-Ray

Example Configuração do coletor de amostras — .yaml otel-collector-config

```
extensions:
  awsproxy:
    endpoint: 127.0.0.1:2000
  health_check:

receivers:
  otlp:
```

```
protocols:
  grpc:
    endpoint: 127.0.0.1:4317
  http:
    endpoint: 127.0.0.1:4318

processors:
  batch:

exporters:
  awsxray:
    region: 'us-east-1'

service:
  pipelines:
    traces:
      receivers: [otlp]
      exporters: [awsxray]
  extensions: [awsproxy, health_check]
```

Important

Configure AWS as credenciais com a `xray:PutTraceSegments` permissão. Para obter mais informações, consulte [Especificar credenciais](#).

Iniciar o coletor

Execute o coletor com seu arquivo de configuração:

```
otelcol --config=otel-collector-config.yaml
```

Migrar no Amazon ECS

Important

Seu perfil de tarefas deve ter a permissão `xray:PutTraceSegments` para qualquer coletor que você usar.

Pare qualquer contêiner X-Ray Daemon existente antes de executar o contêiner do CloudWatch agente ou OpenTelemetry coletor no mesmo host para evitar conflitos de portas.

Usando o CloudWatch agente

1. Obtenha a imagem do Docker na [Galeria Pública do Amazon ECR](#).
2. Crie um arquivo de configuração chamado `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. Armazene a configuração no Systems Manager Parameter Store:

1. Abra as <https://console.aws.amazon.com/systems-manager/>

2. Selecione Criar Parâmetro.

3. Insira os seguintes valores:

- Nome: `/ecs/cwagent/otel-config`
- Nível: Padrão
- Tipo: string
- Tipo de dados: Texto
- Valor: [Cole a configuração `cw-agent-otel.de.json` aqui]

4. Crie uma definição de tarefa usando o modo de rede de ponte:

Na definição da tarefa, a configuração vai depender do modo de rede que você usa. A rede bridge é o padrão e pode ser usado na sua VPC padrão. Em uma rede bridge, defina a variável de `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` ambiente para informar ao OpenTelemetry SDK qual é o endpoint e a porta do CloudWatch agente. Você também deve criar um link do

contêiner do aplicativo para o contêiner do Collector para que os rastreamentos sejam enviados do OpenTelemetry SDK do seu aplicativo para o contêiner do Collector.

Example CloudWatch definição de tarefa do agente

```
{
  "containerDefinitions": [
    {
      "name": "cwagent",
      "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
      "portMappings": [
        {
          "containerPort": 4318,
          "hostPort": 4318,
          "protocol": "tcp"
        },
        {
          "containerPort": 4317,
          "hostPort": 4317,
          "protocol": "tcp"
        },
        {
          "containerPort": 2000,
          "hostPort": 2000,
          "protocol": "tcp"
        }
      ],
      "secrets": [
        {
          "name": "CW_CONFIG_CONTENT",
          "valueFrom": "/ecs/cwagent/otel-config"
        }
      ]
    },
    {
      "name": "application",
      "image": "APPLICATION_IMAGE",
      "links": ["cwagent"],
      "environment": [
        {
          "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
          "value": "http://cwagent:4318/v1/traces"
        }
      ]
    }
  ]
}
```

```

    ]
  }
]
}

```

Para obter mais informações, consulte [Implantação do CloudWatch agente para coletar EC2 métricas em nível de instância da Amazon no Amazon ECS](#).

Usando o OpenTelemetry coletor

1. Obtenha a imagem do Docker `otel/opentelemetry-collector-contrib` do [Docker Hub](#).
2. Crie um arquivo de configuração chamado `otel-collector-config.yaml` usando o mesmo conteúdo mostrado na seção de EC2 configuração do coletor da Amazon, mas atualize os endpoints para usar `0.0.0.0` em vez de `127.0.0.1`.
3. Para usar essa configuração no Amazon ECS, você pode armazenar a configuração no Systems Manager Parameter Store. Primeiro, acesse o console do Systems Manager Parameter Store e selecione Criar novo parâmetro. Crie um parâmetro com as seguintes informações:
 - Nome: `/ecs/otel/config` (esse nome será referenciado na Definição de Tarefa do Coletor)
 - Nível: Padrão
 - Tipo: string
 - Tipo de dados: Texto
 - Valor: [Cole a configuração `otel-collector-config.de.yaml` aqui]
4. Crie uma definição de tarefa para implantar o OpenTelemetry coletor usando o modo de rede bridge como exemplo.

Na definição da tarefa, a configuração vai depender do modo de rede que você usa. A rede bridge é o padrão e pode ser usado na sua VPC padrão. Em uma rede bridge, defina a variável de `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` ambiente para informar ao OpenTelemetry SDK qual é o endpoint e a porta do OpenTelemetry Collector. Você também deve criar um link do contêiner do aplicativo para o contêiner do Collector para que os rastreamentos sejam enviados do OpenTelemetry SDK do seu aplicativo para o contêiner do Collector.

Example OpenTelemetry definição de tarefa de coletor

```

{
  "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"
    }
  ]
}
]
```

Migrar no Elastic Beanstalk

Important

Pare o processo do X-Ray Daemon antes de usar o CloudWatch agente para evitar conflitos de portas.

Sua integração existente do daemon do X-Ray foi ativada usando o console do Elastic Beanstalk ou configurando o daemon do X-Ray no código-fonte da aplicação com um arquivo de configuração.

Usando o CloudWatch agente

Na plataforma Amazon Linux 2, configure o CloudWatch agente usando um arquivo `.ebextensions` de configuração:

1. Crie um diretório chamado `.ebextensions` na raiz do projeto
2. Crie um arquivo denominado `cloudwatch.config` no diretório `.ebextensions` com o conteúdo a seguir:

```
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. Inclua o diretório `.ebextensions` no pacote de fontes da aplicação ao implantar

Para obter mais informações sobre os arquivos de configuração do Elastic Beanstalk, consulte [Personalização avançada de ambiente com arquivos de configuração](#).

Migrando para Java OpenTelemetry

Esta seção fornece orientação sobre como migrar do X-Ray SDK para o OpenTelemetry SDK for Java.

Seções

- [Solução de instrumentação automática de código zero](#)
- [Soluções de instrumentação manual com o SDK](#)
- [Rastrear solicitações recebidas \(instrumentação do Spring Framework\)](#)
- [AWS Instrumentação do SDK v2](#)
- [Instrumentar chamadas HTTP de saída](#)
- [Suporte de instrumentação para outras bibliotecas](#)
- [Criar dados de rastreamento manualmente](#)
- [Instrumentação do Lambda](#)

Solução de instrumentação automática de código zero

With X-Ray Java agent

Para habilitar o agente Java do X-Ray, os argumentos JVM da sua aplicação precisavam ser modificados.

```
-javaagent:/path-to-disco/disco-java-agent.jar=pluginPath=/path-to-disco/disco-plugins
```

With OpenTelemetry-based Java agent

Para usar agentes Java OpenTelemetry baseados.

- Use o agente Java AWS Distro for OpenTelemetry (ADOT) Auto-Instrumentation para instrumentação automática com o agente ADOT Java. Para obter mais informações, consulte

[Auto-Instrumentation for Traces and Metrics with the Java agent](#). Se você quiser somente o rastreamento, desabilite a variável de ambiente `OTEL_METRICS_EXPORTER=none` para exportar métricas do agente Java.

(Opcional) Você também pode ativar o CloudWatch Application Signals ao instrumentar automaticamente seus aplicativos AWS com a instrumentação automática ADOT Java para monitorar a integridade atual do aplicativo e monitorar o desempenho de longo prazo do aplicativo. O Application Signals fornece uma visualização unificada e centrada em aplicações para aplicações, serviços e dependências, além de ajudar você a monitorar e realizar a triagem da integridade da aplicação. Para obter mais informações, consulte [Application Signals](#).

- Use o agente OpenTelemetry Java para instrumentação automática. Para obter mais informações, consulte [Zero-code instrumentation with the Java Agent](#).

Soluções de instrumentação manual com o SDK

Tracing setup with X-Ray SDK

Para instrumentar seu código com o SDK do X-Ray para Java, primeiro era necessário configurar a classe `AWSXRay` com plug-ins de serviço e regras de amostragem locais. Depois, um gravador fornecido era usado.

```
static { AWS XRayRecorderBuilder builder = AWS
XRayRecorderBuilder.standard().withPlugin(new EC2Plugin()).withPlugin(new
ECSPPlugin()); AWS XRay.setGlobalRecorder(builder.build());
}
```

Tracing setup with OpenTelemetry SDK

As dependências a seguir são obrigatórias.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.opentelemetry</groupId>
      <artifactId>opentelemetry-bom</artifactId>
      <version>1.49.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

```

        <groupId>io.opentelemetry.instrumentation</groupId>
        <artifactId>opentelemetry-instrumentation-bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>io.opentelemetry</groupId>
        <artifactId>opentelemetry-sdk</artifactId>
    </dependency>
    <dependency>
        <groupId>io.opentelemetry</groupId>
        <artifactId>opentelemetry-api</artifactId>
    </dependency>
    <dependency>
        <groupId>io.opentelemetry.semconv</groupId>
        <artifactId>opentelemetry-semconv</artifactId>
    </dependency>
    <dependency>
        <groupId>io.opentelemetry</groupId>
        <artifactId>opentelemetry-exporter-otlp</artifactId>
    </dependency>
    <dependency>
        <groupId>io.opentelemetry.contrib</groupId>
        <artifactId>opentelemetry-aws-xray</artifactId>
        <version>1.46.0</version>
    </dependency>
    <dependency>
        <groupId>io.opentelemetry.contrib</groupId>
        <artifactId>opentelemetry-aws-xray-propagator</artifactId>
        <version>1.46.0-alpha</version>
    </dependency>
    <dependency>
        <groupId>io.opentelemetry.contrib</groupId>
        <artifactId>opentelemetry-aws-resources</artifactId>
        <version>1.46.0-alpha</version>
    </dependency>
</dependencies>

```


Configure o OpenTelemetry SDK instanciando um objeto `TracerProvider` e registrando globalmente um objeto. `OpenTelemetrySdk` Configure estes componentes:

- Um exportador OTLP Span (por exemplo, `OtlpGrpcSpanExporter`) - Necessário para exportar traços para o agente ou coletor CloudWatch OpenTelemetry
- Um propagador de AWS raio-X — necessário para propagar o contexto de rastreamento para AWS serviços integrados ao X-Ray
- Um amostrador remoto de AWS raio-X — necessário se você precisar coletar amostras de solicitações usando as regras de amostragem de raios X
- Detectores de recursos (por exemplo, `EcsResource` ou `Ec2Resource`) — Detecte metadados do host que está executando seu aplicativo

```
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();
```

Rastrear solicitações recebidas (instrumentação do Spring Framework)

With X-Ray SDK

Para obter informações sobre como usar o SDK do X-Ray com o Spring Framework para instrumentar sua aplicação, consulte [AOP com Spring e o SDK do X-Ray para Java](#). Para habilitar o AOP no Spring, conclua as etapas a seguir.

1. [Configure o Spring](#)
2. [Adicione um filtro de rastreamento à sua aplicação](#)
3. [Anotar o seu código ou implemente uma interface](#)
4. [Ative o X-Ray em seu aplicativo](#)

With OpenTelemetry SDK

OpenTelemetry fornece bibliotecas de instrumentação para coletar rastreamentos para solicitações recebidas de aplicativos Spring Boot. Para habilitar a instrumentação do Spring Boot com configuração mínima, inclua a dependência a seguir.

```
<dependency>  
    <groupId>io.opentelemetry.instrumentation</groupId>  
    <artifactId>opentelemetry-spring-boot-starter</artifactId>  
</dependency>
```

Para obter mais informações sobre como habilitar e configurar a instrumentação do Spring Boot para sua OpenTelemetry configuração, consulte OpenTelemetry [Introdução](#).

Using OpenTelemetry-based Java agents

O método padrão recomendado para instrumentar aplicativos Spring Boot é usar o [agente OpenTelemetry Java](#) com instrumentação de bytecode, que também fornece mais out-of-the-box instrumentações e configurações quando comparado ao uso direto do SDK. Para começar, consulte [Solução de instrumentação automática de código zero](#).

AWS Instrumentação do SDK v2

With X-Ray SDK

O X-Ray SDK for Java pode instrumentar automaticamente AWS todos os clientes do SDK v2 quando você adiciona `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` o submódulo à sua compilação.

Para instrumentar chamadas posteriores de clientes individuais para AWS serviços com o AWS SDK for Java 2.2 e versões posteriores, `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` o módulo da sua configuração de compilação foi excluído e `aws-xray-recorder-sdk-aws-sdk-v2` o módulo foi incluído. Clientes individuais foram instrumentados configurando-os com um `TracingInterceptor`.

```
import com.amazonaws.xray.interceptors.TracingInterceptor;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
//...

public class MyModel {
    private DynamoDbClient client = DynamoDbClient.builder()
        .region(Region.US_WEST_2)
        .overrideConfiguration(
            ClientOverrideConfiguration.builder()
                .addExecutionInterceptor(new TracingInterceptor())
                .build()
        )
        .build();
//...
```

With OpenTelemetry SDK

Para instrumentar automaticamente todos os clientes AWS do SDK, adicione o `opentelemetry-aws-sdk-2.2-autoconfigure` submódulo.

```
<dependency>
  <groupId>io.opentelemetry.instrumentation</groupId>
  <artifactId>opentelemetry-aws-sdk-2.2-autoconfigure</artifactId>
  <version>2.15.0-alpha</version>
  <scope>runtime</scope>
</dependency>
```

Para instrumentar clientes AWS SDK individuais, adicione o `opentelemetry-aws-sdk-2.2` submódulo.

```
<dependency>
  <groupId>io.opentelemetry.instrumentation</groupId>
  <artifactId>opentelemetry-aws-sdk-2.2</artifactId>
  <version>2.15.0-alpha</version>
  <scope>compile</scope>
</dependency>
```

Em seguida, registre um interceptor ao criar um cliente AWS SDK.

```
import io.opentelemetry.instrumentation.awssdk.v2_2.AwsSdkTelemetry;

// ...

AwsSdkTelemetry telemetry = AwsSdkTelemetry.create(openTelemetry);
private final S3Client S3_CLIENT = S3Client.builder()
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .addExecutionInterceptor(telemetry.newExecutionInterceptor())
        .build())
    .build();
```

Instrumentar chamadas HTTP de saída

With X-Ray SDK

Para instrumentar as solicitações HTTP de saída com o X-Ray, era necessário o X-Ray SDK para Java, versão do Apache HttpClient .

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
public String randomName() throws IOException {
```

```
CloseableHttpClient httpClient = HttpClientBuilder.create().build();
```

With OpenTelemetry SDK

Da mesma forma que o X-Ray Java SDK, OpenTelemetry fornece uma `ApacheHttpClientTelemetry` classe que tem um método construtor que permite a criação de uma instância de um `HttpClientBuilder` para fornecer extensões OpenTelemetry baseadas e propagação de contexto para o Apache. `HttpClient`

```
<dependency>
  <groupId>io.opentelemetry.instrumentation</groupId>
  <artifactId>opentelemetry-apache-httpclient-5.2</artifactId>
  <version>2.15.0-alpha</version>
  <scope>compile</scope>
</dependency>
```

A seguir está um exemplo de código do [opentelemetry-java-instrumentation](#). O cliente HTTP fornecido por `newHttpClient()` gerará rastreamentos para solicitações executadas.

```
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();
    }
}
```

Suporte de instrumentação para outras bibliotecas

Encontre a lista completa de instrumentações de biblioteca suportadas para OpenTelemetry Java em seu respectivo GitHub repositório de instrumentação, em [Bibliotecas, estruturas, servidores](#) de aplicativos e JVMs

Como alternativa, você pode pesquisar no OpenTelemetry Registro para descobrir se OpenTelemetry suporta instrumentação. Para começar a pesquisar, consulte [Registry](#).

Criar dados de rastreamento manualmente

With X-Ray SDK

Com o SDK do X-Ray, os métodos `beginSegment` e `beginSubsegment` são necessários para criar manualmente segmentos e subsegmentos do X-Ray.

```
Segment segment = xrayRecorder.beginSegment("ManualSegment");
segment.putAnnotation("annotationKey", "annotationValue");
segment.putMetadata("metadataKey", "metadataValue");

try {
    Subsegment subsegment =
xrayRecorder.beginSubsegment("ManualSubsegment");
    subsegment.putAnnotation("key", "value");

    // Do something here

} catch (Exception e) {
    subsegment.addException(e);
} finally {
    xrayRecorder.endSegment();
}
```

With OpenTelemetry SDK

Você pode usar extensões personalizadas para monitorar o desempenho de atividades internas que não são capturadas pelas bibliotecas de instrumentação. Somente extensões do tipo servidor

são convertidas em segmentos do X-Ray. Todas as outras extensões são convertidas em subsegmentos do X-Ray.

Primeiro, você precisará criar um Rastreador para gerar extensões, que podem ser obtidas por meio do método `openTelemetry.getTracer`. Isso fornecerá uma instância do rastreador de `TracerProvider` que foi registrada globalmente no exemplo [Soluções de instrumentação manual com o SDK](#). Você pode criar quantas instâncias de rastreador forem necessárias, mas é comum ter um rastreador para toda a aplicação.

```
Tracer tracer = openTelemetry.getTracer("my-app");
```

Você pode usar o rastreador para criar extensões.

```
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();
```

As extensões têm um tipo padrão `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();
}
```

Adicionar anotações e metadados aos rastreamentos com o SDK OpenTelemetry

No exemplo acima, o método `setAttribute` é usado para adicionar atributos a cada extensão. Por padrão, todos os atributos de extensão serão convertidos em metadados nos dados brutos do X-Ray. Para garantir que um atributo seja convertido em uma anotação, e não em metadados, o exemplo acima adiciona a chave desse atributo à lista do atributo `aws.xray.annotations`. Para obter mais informações, consulte [Enable The Customized X-Ray Annotations](#) e [Anotações e metadados](#).

Com agentes Java OpenTelemetry baseados

Se estiver usando o agente Java para instrumentar automaticamente a aplicação, você precisará realizar a instrumentação manual na aplicação. Por exemplo, para instrumentar o código dentro da aplicação para seções que não são cobertas por nenhuma biblioteca de instrumentação automática.

Você precisa usar o artefato `opentelemetry-api` para realizar a instrumentação manual com o agente. A versão do artefato não pode ser mais recente que a versão do agente.

```
import io.opentelemetry.api.GlobalOpenTelemetry;
import io.opentelemetry.api.trace.Span;

// ...

Span parentSpan = Span.current();
Tracer tracer = GlobalOpenTelemetry.getTracer("my-app");
Span span = tracer.spanBuilder("my-span-name")
    .setParent(io.opentelemetry.context.Context.current().with(parentSpan))
    .startSpan();
span.end();
```


Instrumentação do Lambda

With X-Ray SDK

Ao usar o SDK do X-Ray, depois que o Lambda tiver o Rastreamento ativo habilitado, não há necessidade de configuração adicional para usar o SDK do X-Ray. O Lambda criará um segmento representando a invocação do manipulador do Lambda, e você poderá criar subsegmentos ou bibliotecas de instrumentos usando o SDK do X-Ray sem nenhuma configuração adicional.

With OpenTelemetry-based solutions

Camadas Lambda de instrumentação automática — Você pode instrumentar automaticamente seu Lambda AWS com camadas Lambda vendidas usando as seguintes soluções:

- CloudWatch Camada Lambda de sinais de aplicação (recomendada)



Note

Essa camada Lambda tem sinais de CloudWatch aplicativo ativados por padrão, o que permite o monitoramento de desempenho e integridade de seu aplicativo Lambda por meio da coleta de métricas e rastreamentos. Para usar somente o rastreamento, defina a variável de ambiente `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false` do Lambda.

- Permite o monitoramento do desempenho e da integridade da sua aplicação Lambda
- Coleta métricas e rastreamentos por padrão
- AWS camada Lambda gerenciada para ADOT Java. Para obter mais informações, consulte [AWS Distro for OpenTelemetry Lambda Support For Java](#).

Para usar a instrumentação manual com a camada de instrumentação automática, consulte [Soluções de instrumentação manual com o SDK](#). Para reduzir as partidas a frio, considere usar instrumentação OpenTelemetry manual para gerar OpenTelemetry traços para sua função Lambda.

OpenTelemetry instrumentação manual para Lambda AWS

Considere o seguinte código de função Lambda que faz uma chamada do Amazon ListBuckets S3.

```
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);
        }
    }
}
```

Confira as dependências.

```
dependencies {  
    implementation('com.amazonaws:aws-lambda-java-core:1.2.3')  
    implementation('software.amazon.awssdk:s3:2.28.29')  
    implementation('org.slf4j:slf4j-nop:2.0.16')  
}
```

Para instrumentar manualmente seu manipulador do Lambda e o cliente do Amazon S3, faça o seguinte:

1. Substitua suas classes de função que implementam `RequestHandler` (ou `RequestStreamHandler`) por aquelas que se estendem `TracingRequestHandler` (ou `TracingRequestStreamHandler`).
2. `TracerProvider` Instancie um objeto e registre globalmente. `OpenTelemetrySdk TracerProvider` Recomenda-se que seja configurado com:
 - a. Um processador de extensões simples com um exportador de extensões do UDP do X-Ray para enviar rastreamentos ao endpoint UDP X-Ray do Lambda
 - b. Um amostrador `ParentBased` sempre ativo (padrão se não estiver configurado)
 - c. Um recurso com `service.name` definido como o nome da função do Lambda
 - d. Um propagador do X-Ray Lambda
3. Altere o método `handleRequest` para `doHandleRequest` e transmita o objeto `OpenTelemetrySdk` para a classe de base.
4. Instrumente o cliente Amazon S3 com a instrumentação do OpenTemetry AWS SDK registrando o interceptor ao criar o cliente.

Você precisa das seguintes dependências `OpenTelemetry` relacionadas.

```
dependencies {  
    ...  
  
    implementation("software.amazon.distro.opentelemetry:aws-distro-opentelemetry-xray-  
udp-span-exporter:0.1.0")  
  
    implementation(platform('io.opentelemetry.instrumentation:opentelemetry-  
instrumentation-bom:2.14.0'))  
    implementation(platform('io.opentelemetry:opentelemetry-bom:1.48.0'))  
  
    implementation('io.opentelemetry:opentelemetry-sdk')
```

```

    implementation('io.opentelemetry:opentelemetry-api')
    implementation('io.opentelemetry.contrib:opentelemetry-aws-xray-propagator:1.45.0-alpha')
    implementation('io.opentelemetry.contrib:opentelemetry-aws-resources:1.45.0-alpha')
    implementation('io.opentelemetry.instrumentation:opentelemetry-aws-lambda-core-1.0:2.14.0-alpha')
    implementation('io.opentelemetry.instrumentation:opentelemetry-aws-sdk-2.2:2.14.0-alpha')
}

```

O código a seguir demonstra a função do Lambda após as alterações necessárias. Você pode criar extensões personalizadas adicionais para complementar as extensões fornecidas automaticamente.

```

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);
    }
}

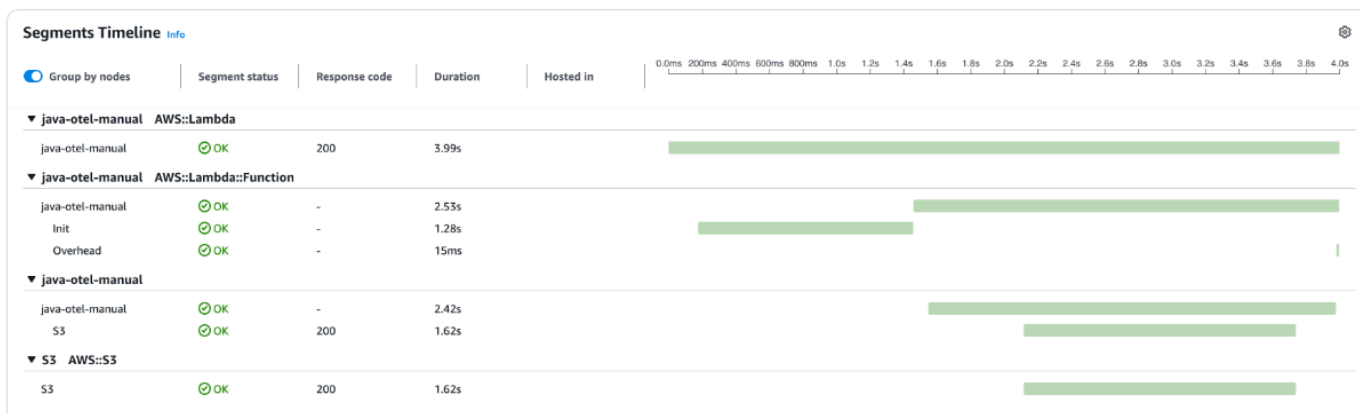
```

```

    return response;
  } catch (S3Exception e) {
    throw new RuntimeException("Failed to call S3 listBuckets" +
      e.awsErrorDetails().errorMessage(), e);
  }
}
}

```

Ao invocar a função Lambda, você verá o seguinte rastreamento em Trace Map no CloudWatch console.



Migre para Go OpenTelemetry

Use os exemplos de código a seguir para instrumentar manualmente seus aplicativos Go com o OpenTelemetry SDK ao migrar do X-Ray.

Instrumentação manual com o SDK

Tracing setup with X-Ray SDK

Ao usar o SDK do X-Ray para Go, era necessário configurar plug-ins de serviço ou regras de amostragem locais antes de instrumentar seu código.

```
func init() {
    if os.Getenv("ENVIRONMENT") == "production" {
        ec2.Init()
    }

    xray.Configure(xray.Config{
        DaemonAddr:      "127.0.0.1:2000",
        ServiceVersion:   "1.2.3",
    })
}
```

Set up tracing with OpenTelemetry SDK

Configure o OpenTelemetry SDK instanciando um TracerProvider e registrando-o como provedor global de rastreadores. É recomendável configurar os seguintes componentes:

- Exportador de rastreamento OTLP — Necessário para exportar traços para o agente ou coletor CloudWatch OpenTelemetry
- X-Ray Propagator — necessário para propagar o contexto de rastreamento para AWS serviços integrados ao X-Ray
- Amostrador remoto do X-Ray: necessário para solicitações de amostragem usando regras de amostragem do X-Ray
- Detectores de recursos: para detectar metadados do host que está executando sua aplicação

```
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
}
```

Rastrear solicitações de entrada (instrumentação do manipulador HTTP)

With X-Ray SDK

Para instrumentar um manipulador HTTP com X-Ray, o método do manipulador X-Ray foi usado para gerar segmentos usando `NewFixedSegmentNamer`


```
func main() {
    http.Handle("/", xray.Handler(xray.NewFixedSegmentNamer("myApp"),
    http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Hello!"))
    })))
    http.ListenAndServe(":8000", nil)
}
```

With OpenTelemetry SDK

Para instrumentar um manipulador HTTP com OpenTelemetry, use o OpenTelemetry método `newHandler` para encapsular o código original do manipulador.

```
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 para instrumentação do SDK for Go v2

With X-Ray SDK

Para instrumentar as AWS solicitações de saída do AWS SDK, seus clientes foram instrumentados da seguinte forma:

```
// 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.AWSSDKV2Instrumentor(&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

O suporte de rastreamento para chamadas downstream do AWS SDK é fornecido pela instrumentação OpenTelemetry do SDK AWS for Go v2. Veja um exemplo de rastreamento de uma chamada do cliente do S3:

```
import (
    ...

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"

    "go.opentelemetry.io/otel"
```

```
oteltrace "go.opentelemetry.io/otel/trace"
awsConfig "github.com/aws/aws-sdk-go-v2/config"
"go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-sdk-go-v2/
otelaws"
)

...

// init aws config
cfg, err := awsConfig.LoadDefaultConfig(ctx)
if err != nil {
    panic("configuration error, " + err.Error())
}

// instrument all aws clients
otelaws.AppendMiddlewares(&.APIOptions)

// Call to S3
s3Client := s3.NewFromConfig(cfg)
input := &s3.ListBucketsInput{}
result, err := s3Client.ListBuckets(ctx, input)
if err != nil {
    fmt.Printf("Got an error retrieving buckets, %v", err)
    return
}
```

Instrumentar chamadas HTTP de saída

With X-Ray SDK

Para instrumentar as chamadas HTTP de saída com o X-Ray, o `xray.Client` foi usado para criar uma cópia de um cliente HTTP fornecido.

```
myClient := xray.Client(http-client)

resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

With OpenTelemetry SDK

Para instrumentar clientes HTTP com OpenTelemetry, use OpenTelemetry otelhttp. NewTransport método para encapsular o http. DefaultTransport.

```
import (
    "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"
)

// Create an instrumented HTTP client.
httpClient := &http.Client{
    Transport: otelhttp.NewTransport(
        http.DefaultTransport,
    ),
}

req, err := http.NewRequestWithContext(ctx, http.MethodGet, "https://api.github.com/
repos/aws-observability/aws-otel-go/releases/latest", nil)
if err != nil {
    fmt.Printf("failed to create http request, %v\n", err)
}
res, err := httpClient.Do(req)
if err != nil {
    fmt.Printf("failed to make http request, %v\n", err)
}
// Request body must be closed
defer func(Body io.ReadCloser) {
    err := Body.Close()
    if err != nil {
        fmt.Printf("failed to close http response body, %v\n", err)
    }
}(res.Body)
```

Suporte de instrumentação para outras bibliotecas

Você pode encontrar a lista completa de instrumentações de biblioteca compatíveis com o OpenTelemetry Go em Pacotes de [instrumentação](#).

[Como alternativa, você pode pesquisar no OpenTelemetry registro para descobrir se OpenTelemetry suporta instrumentação para sua biblioteca em Registro.](#)

Criar dados de rastreamento manualmente

With X-Ray SDK

Com o X-Ray SDK, os `BeginSubsegment` métodos `BeginSegment` e foram necessários para criar manualmente segmentos e subsegmentos de raio-X.

```
// Start a segment
ctx, seg := xray.BeginSegment(context.Background(), "service-name")
// Start a subsegment
subCtx, subSeg := xray.BeginSubsegment(ctx, "subsegment-name")

// Add metadata or annotation here if necessary
xray.AddAnnotation(subCtx, "annotationKey", "annotationValue")
xray.AddMetadata(subCtx, "metadataKey", "metadataValue")

subSeg.Close(nil)
// Close the segment
seg.Close(nil)
```

With OpenTelemetry SDK

Use extensões personalizadas para monitorar o desempenho de atividades internas que não são capturadas pelas bibliotecas de instrumentação. Somente extensões do tipo servidor são convertidas em segmentos do X-Ray. Todas as outras extensões são convertidas em subsegmentos do X-Ray.

Primeiro, você precisará criar um rastreador para gerar extensões, que podem ser obtidas por meio do método `otel.Tracer`. Isso fornecerá uma instância do `Tracer TracerProvider` que foi registrada globalmente no exemplo de configuração de rastreamento. Você pode criar quantas instâncias de rastreador forem necessárias, mas é comum ter um rastreador para toda a aplicação.

```
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()

```

Adicionar anotações e metadados aos rastreamentos com o SDK OpenTelemetry

No exemplo acima, o método `WithAttributes` é usado para adicionar atributos a cada extensão. Por padrão, todos os atributos de extensão são convertidos em metadados nos dados brutos do X-Ray. Para garantir que um atributo seja convertido em uma anotação, e não em metadados, adicione a chave do atributo à lista do atributo `aws.xray.annotations`. Para obter mais informações, consulte [Enable The Customized X-Ray Annotations](#).

Instrumentação manual Lambda

With X-Ray SDK

Ao usar o SDK do X-Ray, depois que o Lambda tiver o Rastreamento ativo habilitado, não há necessidade de configurações adicionais para usar o SDK do X-Ray. O Lambda criou um

segmento representando a invocação do manipulador do Lambda, e você criou subsegmentos usando o SDK do X-Ray sem nenhuma configuração adicional.

With OpenTelemetry SDK

O código de função Lambda a seguir (sem instrumentação) faz uma chamada do Amazon S3 e uma solicitação HTTP de saída ListBuckets .

```
package main

import (
    "context"
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "os"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    awsconfig "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func lambdaHandler(ctx context.Context) (interface{}, error) {
    // Initialize AWS config.
    cfg, err := awsconfig.LoadDefaultConfig(ctx)
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    s3Client := s3.NewFromConfig(cfg)

    // Create an HTTP client.
    httpClient := &http.Client{
        Transport: http.DefaultTransport,
    }

    input := &s3.ListBucketsInput{}
    result, err := s3Client.ListBuckets(ctx, input)
    if err != nil {
        fmt.Printf("Got an error retrieving buckets, %v", err)
    }
}
```

```

    fmt.Println("Buckets:")
    for _, bucket := range result.Buckets {
        fmt.Println(*bucket.Name + ": " + bucket.CreationDate.Format("2006-01-02
15:04:05 Monday"))
    }
    fmt.Println("End Buckets.")

    req, err := http.NewRequestWithContext(ctx, http.MethodGet, "https://
api.github.com/repos/aws-observability/aws-otel-go/releases/latest", nil)
    if err != nil {
        fmt.Printf("failed to create http request, %v\n", err)
    }
    res, err := httpClient.Do(req)
    if err != nil {
        fmt.Printf("failed to make http request, %v\n", err)
    }
    defer func(Body io.ReadCloser) {
        err := Body.Close()
        if err != nil {
            fmt.Printf("failed to close http response body, %v\n", err)
        }
    }(res.Body)

    var data map[string]interface{}
    err = json.NewDecoder(res.Body).Decode(&data)
    if err != nil {
        fmt.Printf("failed to read http response body, %v\n", err)
    }
    fmt.Printf("Latest ADOT Go Release is '%s'\n", data["name"])

    return events.APIGatewayProxyResponse{
        StatusCode: http.StatusOK,
        Body:       os.Getenv("_X_AMZN_TRACE_ID"),
    }, nil
}

func main() {
    lambda.Start(lambdaHandler)
}

```


Para instrumentar manualmente seu manipulador do Lambda e o cliente do Amazon S3, faça o seguinte:

1. Em `main()`, instancie um `TracerProvider` (`tp`) e registre-o como o provedor global de rastreadores. `TracerProvider` Recomenda-se que seja configurado com:
 - a. Processador de extensão simples com um exportador de extensão UDP X-Ray para enviar rastreamentos ao endpoint UDP X-Ray do Lambda
 - b. Recurso com `service.name` definido como o nome da função do Lambda
2. Altere o uso de `lambda.Start(lambdaHandler)` para `lambda.Start(otellambda.InstrumentHandler(lambdaHandler, xrayconfig.WithRecommendedOptions(tp)...))`.
3. Instrumente o cliente Amazon S3 com a instrumentação do OpenTelemetry AWS SDK anexando OpenTelemetry middleware à configuração do cliente Amazon S3. `aws-sdk-go-v2`
4. Instrumente o cliente http usando o `otelhttp.NewTransport` método OpenTelemetry's para encapsular `http.DefaultTransport` o.

O código a seguir é um exemplo de como a função do Lambda ficará após as alterações. Você pode criar manualmente extensões personalizadas adicionais, além das extensões fornecidas automaticamente.

```
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(otelambda.InstrumentHandler(lambdaHandler,
xrayconfig.WithRecommendedOptions(tp)...))
}
```

Ao invocar o Lambda, você verá o seguinte rastreamento Trace Map no CloudWatch console:



Migrar para Node.js OpenTelemetry

Esta seção explica como migrar seus aplicativos Node.js do X-Ray SDK para o OpenTelemetry. Ela abrange abordagens de instrumentação automática e manual e fornece exemplos específicos para casos de uso comuns.

O SDK X-Ray Node.js ajuda você a instrumentar manualmente suas aplicações Node.js para rastreamento. Esta seção fornece exemplos de código para migrar do X-Ray para a OpenTelemetry instrumentação.

Seções

- [Soluções de instrumentação automática de código zero](#)
- [Soluções de instrumentação manual](#)
- [Rastreamento de solicitações de entrada](#)
- [AWS Instrumentação do SDK JavaScript V3](#)
- [Instrumentar chamadas HTTP de saída](#)
- [Suporte de instrumentação para outras bibliotecas](#)
- [Criar dados de rastreamento manualmente](#)
- [Instrumentação do Lambda](#)

Soluções de instrumentação automática de código zero

Para rastrear solicitações com o SDK do X-Ray para Node.js, você deve modificar o código da aplicação. Com OpenTelemetry, você pode usar soluções de instrumentação automática de código zero para rastrear solicitações.

Instrumentação automática de código zero com instrumentações automáticas OpenTelemetry baseadas.

1. Usando a AWS distro para instrumentação automática OpenTelemetry (ADOT) para Node.js — Para instrumentação automática para o aplicativo Node.js, consulte [Rastreamento e métricas com a distribuição para instrumentação](#) automática. AWS OpenTelemetry JavaScript

(Opcional) Você também pode ativar o CloudWatch Application Signals ao instrumentar automaticamente seus aplicativos AWS com a JavaScript instrumentação automática ADOT para monitorar a integridade atual do aplicativo e acompanhar o desempenho de longo prazo

do aplicativo em relação aos seus objetivos de negócios. O Application Signals fornece uma visualização unificada e centrada em aplicativos para aplicativos, serviços e dependências, além de ajudar você a monitorar e realizar a triagem da integridade do aplicativo. Para obter mais informações, consulte [Application Signals](#).

2. [Usando a instrumentação automática de OpenTelemetry JavaScript código zero — Para instrumentação automática com o, consulte instrumentação de código zero. OpenTelemetry JavaScript JavaScript](#)

Soluções de instrumentação manual

Tracing setup with X-Ray SDK

Quando o SDK do X-Ray para Node.js foi usado, o pacote `aws-xray-sdk` precisou configurar o SDK do X-Ray com plug-ins de serviço ou regras de amostragem locais antes de usar o SDK para instrumentar seu código.

```
var AWSXRay = require('aws-xray-sdk');

AWSXRay.config([AWSXRay.plugins.EC2Plugin, AWSXRay.plugins.ElasticBeanstalkPlugin]);
AWSXRay.middleware.setSamplingRules(<path to file>);
```

Tracing setup with OpenTelemetry SDK

Note

AWS No momento, o X-Ray Remote Sampling não está disponível para ser configurado para OpenTelemetry JS. No entanto, o suporte para a amostragem remota do X-Ray está atualmente disponível por meio da instrumentação automática do ADOT para Node.js.

Para o exemplo de código abaixo, você precisará das seguintes dependências:

```
npm install --save \
  @opentelemetry/api \
  @opentelemetry/sdk-node \
  @opentelemetry/exporter-trace-otlp-proto \
```

```
@opentelemetry/propagator-aws-xray \  
@opentelemetry/resource-detector-aws
```

Você deve instalar e configurar o OpenTelemetry SDK antes de executar o código do aplicativo. Isso pode ser feito usando o sinalizador [—require](#). Crie um arquivo chamado `instrumentation.js`, que conterá sua configuração e configuração OpenTelemetry de instrumentação.

É recomendável configurar os seguintes componentes:

- **OTLPTraceExportador** — Obrigatório para exportar traços para o Agente/Coletor CloudWatch OpenTelemetry
- **AWSXRayPropagator** — Necessário para propagar o Trace Context para AWS serviços integrados ao X-Ray
- **Detectores de recursos** (por exemplo, Amazon EC2 Resource Detector) - Para detectar metadados do host que executa seu aplicativo

```
/*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();
```

Em seguida, você pode executar seu aplicativo com sua OpenTelemetry configuração, como:

```
node --require ./instrumentation.js app.js
```

Você pode usar as instrumentações da biblioteca do OpenTelemetry SDK para criar automaticamente extensões para bibliotecas como o SDK. AWS Habilitá-los criará automaticamente extensões para módulos, como o AWS SDK para JavaScript v3. OpenTelemetry fornece a opção de ativar todas as instrumentações da biblioteca ou especificar quais instrumentações da biblioteca devem ser ativadas.

Para habilitar todas as instrumentações, instale o pacote `@opentelemetry/auto-instrumentations-node`:

```
npm install @opentelemetry/auto-instrumentations-node
```

Em seguida, atualize a configuração para habilitar todas as instrumentações de biblioteca, conforme mostrado abaixo.

```
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

Você pode usar a instrumentação automática ADOT para Node.js OpenTelemetry para configurar automaticamente seus aplicativos Node.js. Ao usar o ADOT Auto-Instrumentation,

you don't need to make manual changes to the code to trace requests received or trace libraries, such as the AWS SDK or HTTP. For more information, consult [Rastreamento e métricas com a AWS distribuição para OpenTelemetry JavaScript](#) automatic instrumentation.

Automatic instrumentation of ADOT for Node.js is compatible with:

- Remote sampling of X-Ray by means of environment variable: `export OTEL_TRACES_SAMPLER=xray`
- Propagation of the context of X-Ray tracing (enabled by default)
- Detection of resources (the detection of resources for Amazon EC2, Amazon ECS and Amazon EKS is activated by default)
- Automatic instrumentation libraries for all OpenTelemetry instrumentations supported, which can be disabled/enabled selectively by means of environment variables `OTEL_NODE_ENABLED_INSTRUMENTATIONS` `OTEL_NODE_DISABLED_INSTRUMENTATIONS`
- Manual creation of expansions

Rastreamento de solicitações de entrada

With X-Ray SDK

Express.js

With the X-Ray SDK to trace HTTP requests received by applications Express.js, the two middlewares `AWSXRay.express.openSegment(<name>)` and `AWSXRay.express.closeSegment()` needed to group all the defined routes to trace them.

```
app.use(xrayExpress.openSegment('defaultName'));  
  
...  
  
app.use(xrayExpress.closeSegment());
```

Restify

Para rastrear as solicitações HTTP recebidas pelas aplicações Restify, o middleware do SDK do X-Ray foi usado executando `enable` no módulo `aws-xray-sdk-restify` no servidor Restify:

```
var AWSXRay = require('aws-xray-sdk');
var AWSXRayRestify = require('aws-xray-sdk-restify');

var restify = require('restify');
var server = restify.createServer();
AWSXRayRestify.enable(server, 'MyApp');
```

With OpenTelemetry SDK

Express.js

[O suporte de rastreamento para solicitações recebidas Express.js é fornecido pela instrumentação OpenTelemetry HTTP e pela instrumentação expressa. OpenTelemetry](#) Instale as seguintes dependências com npm:

```
npm install --save @opentelemetry/instrumentation-http @opentelemetry/
instrumentation-express
```

Atualize a configuração do OpenTelemetry SDK para habilitar a instrumentação para o módulo expresso:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
const { ExpressInstrumentation } = require('@opentelemetry/instrumentation-express');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    // Express instrumentation requires HTTP instrumentation
    new HttpInstrumentation(),
```

```
    new ExpressInstrumentation(),  
  ],  
});
```

Restify

Para aplicativos Restify, você precisará da instrumentação [OpenTelemetry Restify](#). Instale a seguinte dependência:

```
npm install --save @opentelemetry/instrumentation-restify
```

Atualize a configuração do OpenTelemetry SDK para habilitar a instrumentação para o módulo restify:

```
const { RestifyInstrumentation } = require('@opentelemetry/instrumentation-restify');  
...  
  
const sdk = new NodeSDK({  
  ...  
  
  instrumentations: [  
    ...  
    new RestifyInstrumentation(),  
  ],  
});
```

AWS Instrumentação do SDK JavaScript V3

With X-Ray SDK

Para instrumentar as AWS solicitações de saída do AWS SDK, você instrumentou clientes como no exemplo a seguir:

```
import { S3, PutObjectCommand } from '@aws-sdk/client-s3';

const s3 = AWSXRay.captureAWSSv3Client(new S3({}));

await s3.send(new PutObjectCommand({
  Bucket: bucketName,
  Key: keyName,
  Body: 'Hello!',
}));
```

With OpenTelemetry SDK

O suporte de rastreamento para chamadas downstream do AWS SDK para o DynamoDB, o Amazon S3 e outros é fornecido pela instrumentação do SDK. OpenTelemetry AWS Instale a seguinte dependência com npm:

```
npm install --save @opentelemetry/instrumentation-aws-sdk
```

Atualize a configuração do OpenTelemetry SDK com a instrumentação do AWS SDK.

```
import { AwsInstrumentation } from '@opentelemetry/instrumentation-aws-sdk';
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new AwsInstrumentation()
  ],
});
```

Instrumentar chamadas HTTP de saída

With X-Ray SDK

Para instrumentar solicitações HTTP de saída com o X-Ray, era necessário instrumentar os clientes. Confira um exemplo abaixo.

Clientes HTTP individuais

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Todos os clientes HTTP (globais)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

With OpenTelemetry SDK

O suporte de rastreamento para clientes HTTP Node.js é fornecido pela Instrumentação OpenTelemetry HTTP. Instale a seguinte dependência com npm:

```
npm install --save @opentelemetry/instrumentation-http
```

Atualize a configuração do OpenTelemetry SDK da seguinte forma:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new HttpInstrumentation(),
  ],
```

```
});
```

Suporte de instrumentação para outras bibliotecas

Você pode encontrar a lista completa de instrumentações de biblioteca suportadas OpenTelemetry JavaScript em Instrumentações [suportadas](#).

[Como alternativa, você pode pesquisar no OpenTelemetry registro para descobrir se OpenTelemetry suporta instrumentação para sua biblioteca em Registro.](#)

Criar dados de rastreamento manualmente

With X-Ray SDK

Ao usar o X-Ray, o código do pacote `aws-xray-sdk` era necessário para criar manualmente segmentos e seus subsegmentos filho para rastrear sua aplicação.

```
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

Você pode criar e usar extensões personalizadas para monitorar o desempenho de atividades internas que não são capturadas pelas bibliotecas de instrumentação. Somente extensões do tipo servidor são convertidas em segmentos do X-Ray. Todas as outras extensões são convertidas em subsegmentos do X-Ray. Para mais informações, leia [Segmentos](#).

Você precisará de uma instância do Tracer depois de configurar o OpenTelemetry SDK na Configuração de rastreamento para criar Spans. Você pode criar quantas instâncias de rastreador forem necessárias, mas é comum ter um rastreador para toda a aplicação.

```
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();
});
```

Adicionar anotações e metadados aos rastreamentos com o SDK OpenTelemetry

Você também pode adicionar pares de valor-chave personalizados como atributos nas extensões. Por padrão, todos esses atributos de extensão serão convertidos em metadados nos dados brutos do X-Ray. Para garantir que um atributo seja convertido em uma anotação, e não em metadados, adicione a chave do atributo à lista do atributo `aws.xray.annotations`. Para obter mais informações, consulte [Enable The Customized X-Ray Annotations](#).

```
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();
});
```

Instrumentação do Lambda

With X-Ray SDK

Após a habilitação do Rastreamento ativo para sua função do Lambda, o SDK do X-Ray será necessário sem configuração adicional. O Lambda cria um segmento representando a invocação do manipulador do Lambda, e você cria subsegmentos ou bibliotecas de instrumentos usando o SDK do X-Ray sem nenhuma configuração adicional.

With OpenTelemetry SDK

Você pode instrumentar automaticamente seu Lambda com camadas personalizadas do AWS Lambda. Existem duas soluções:

- Camada lambda (recomendada) CloudWatch do Application Signals

Note

Essa camada Lambda tem sinais de CloudWatch aplicativo ativados por padrão, o que permite o monitoramento de desempenho e integridade de seu aplicativo Lambda por meio da coleta de métricas e rastreamentos. Se você quiser somente o rastreamento, defina a variável de ambiente `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false` do Lambda. Para obter mais informações, consulte [Habilitar aplicações no Lambda](#).

- AWS camada Lambda gerenciada para ADOT JS. Para obter mais informações, consulte [AWS Distro for OpenTelemetry Lambda Support JavaScript For](#).

Criar manualmente extensões com instrumentação do Lambda

Embora o ADOT JavaScript Lambda Layer forneça instrumentação automática para sua função Lambda, você pode achar necessário realizar instrumentação manual em seu Lambda, por

exemplo, para fornecer dados personalizados ou instrumentar código dentro da própria função Lambda que não seja coberto pelas instrumentações da biblioteca.

Para realizar a instrumentação manual com a instrumentação automática, você precisará adicionar `@opentelemetry/api` como uma dependência. Recomenda-se que a versão dessa dependência seja a mesma versão da mesma dependência usada pelo SDK JavaScript ADOT. Você pode usar a OpenTelemetry API para criar extensões manualmente em sua função Lambda.

Para adicionar a dependência `@opentelemetry/api` usando o NPM:

```
npm install @opentelemetry/api
```

Migrar para o.NET OpenTelemetry

Ao usar o rastreamento do X-Ray nas suas aplicações .NET, o SDK .NET do X-Ray com esforços manuais é usado para instrumentação.

Esta seção fornece exemplos de código na [Soluções de instrumentação manual com o SDK](#) seção para migrar da solução de instrumentação manual X-Ray para soluções de instrumentação OpenTelemetry manual para.NET. Como alternativa, você pode migrar da instrumentação manual X-Ray para soluções de instrumentação OpenTelemetry automática para instrumentar aplicativos.NET sem precisar modificar o código-fonte do aplicativo na seção. [Soluções de instrumentação automática de código zero](#)

Seções

- [Soluções de instrumentação automática de código zero](#)
- [Soluções de instrumentação manual com o SDK](#)
- [Criar dados de rastreamento manualmente](#)
- [Rastrear solicitações recebidas \(instrumentação do ASP.NET e do ASP.NET Core\)](#)
- [AWS Instrumentação do SDK](#)
- [Instrumentar chamadas HTTP de saída](#)
- [Suporte de instrumentação para outras bibliotecas](#)
- [Instrumentação do Lambda](#)

Soluções de instrumentação automática de código zero

OpenTelemetry fornece soluções de instrumentação automática de código zero. Essas soluções rastreiam solicitações sem exigir alterações no código da aplicação.

OpenTelemetry oferece opções de instrumentação automática baseadas em

1. Usando a AWS distro para instrumentação automática OpenTelemetry (ADOT) para .NET — Para instrumentar automaticamente aplicativos .NET, consulte [Rastreamento e métricas com a distribuição para instrumentação automática .NET](#). AWS OpenTelemetry

(Opcional) Ative os sinais de CloudWatch aplicativos ao instrumentar automaticamente seus aplicativos AWS com a instrumentação automática ADOT .NET para:

- Monitorar a integridade atual da aplicação
- Acompanhar o desempenho a longo prazo da aplicação em relação aos objetivos de negócios
- Obter uma visualização unificada e centrada em aplicações das suas aplicações, serviços e dependências
- Monitorar e fazer a triagem da integridade da aplicação

Para obter mais informações, consulte [Application Signals](#).

2. Usando a instrumentação automática de código zero OpenTelemetry do .NET — Para instrumentar automaticamente com OpenTelemetry o .NET, consulte [Rastreamento e métricas com a distribuição para instrumentação automática do .NET](#). AWS OpenTelemetry

Soluções de instrumentação manual com o SDK

Tracing configuration with X-Ray SDK

Para aplicativos web .NET, o SDK do X-Ray é configurado na seção `appSettings` do arquivo `Web.config`.

Exemplo de `Web.config`

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
  </appSettings>
</configuration>
```

Para o .NET Core, um arquivo chamado `appsettings.json` com uma chave de nível superior chamada `XRay` é usado e, em seguida, um objeto de configuração é criado para inicializar o gravador do X-Ray.

Exemplo de .NET `appsettings.json`

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin"
  }
}
```

Exemplo de .NET Core `Program.cs`: configuração do gravador

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Tracing configuration with OpenTelemetry SDK

Adicione estas dependências:

```
dotnet add package OpenTelemetry
dotnet add package OpenTelemetry.Contrib.Extensions.AWSXRay
dotnet add package OpenTelemetry.Sampler.AWS --prerelease
dotnet add package OpenTelemetry.Resources.AWS
dotnet add package OpenTelemetry.Exporter.OpenTelemetryProtocol
dotnet add package OpenTelemetry.Extensions.Hosting
dotnet add package OpenTelemetry.Instrumentation.AspNetCore
```

Para seu aplicativo .NET, configure o OpenTelemetry SDK configurando o `Global TracerProvider`. O exemplo de configuração a seguir também permite a instrumentação de `ASP.NET Core`. Para

instrumentar ASP.NET, consulte [Rastrear solicitações recebidas \(instrumentação do ASP.NET e do ASP.NET Core\)](#). Para usar OpenTelemetry com outras estruturas, consulte [Registro](#) para obter mais bibliotecas de estruturas compatíveis.

É recomendável configurar os seguintes componentes:

- An OTLP Exporter—Necessário para exportar traços para o CloudWatch OpenTelemetry Agente/Coletor
- Um propagador de AWS raio-X — necessário para propagar o contexto de rastreamento para [AWS serviços integrados](#) ao X-Ray
- Um amostrador remoto de AWS raio-X — necessário se você precisar coletar amostras de [solicitações usando as regras de amostragem de raio-X](#)
- Resource Detectors(por exemplo, Amazon EC2 Resource Detector) - Para detectar metadados do host que executa seu aplicativo

```
using OpenTelemetry;
using OpenTelemetry.Contrib.Extensions.AWSXRay.Trace;
using OpenTelemetry.Sampler.AWS;
using OpenTelemetry.Trace;
using OpenTelemetry.Resources;

var builder = WebApplication.CreateBuilder(args);

var serviceName = "MyServiceName";
var serviceVersion = "1.0.0";

var resourceBuilder = ResourceBuilder
    .CreateDefault()
    .AddService(serviceName: serviceName)
    .AddAWSEC2Detector();

builder.Services.AddOpenTelemetry()
    .ConfigureResource(resource => resource
        .AddAWSEC2Detector()
        .AddService(
            serviceName: serviceName,
            serviceVersion: serviceVersion))
    .WithTracing(tracing => tracing
        .AddSource(serviceName)
```

```
.AddAspNetCoreInstrumentation()  
.AddOtlpExporter()  
.SetSampler(AWSXRayRemoteSampler.Builder(resourceBuilder.Build())  
    .SetEndpoint("http://localhost:2000")  
    .Build()));
```

```
Sdk.SetDefaultTextMapPropagator(new AWSXRayPropagator()); // configure X-Ray  
propagator
```

OpenTelemetry Para usar em um aplicativo de console, adicione a seguinte OpenTelemetry configuração na inicialização do seu programa.

```
using OpenTelemetry;  
using OpenTelemetry.Contrib.Extensions.AWSXRay.Trace;  
using OpenTelemetry.Trace;  
using OpenTelemetry.Resources;  
  
var serviceName = "MyServiceName";  
  
var resourceBuilder = ResourceBuilder  
    .CreateDefault()  
    .AddService(serviceName: serviceName)  
    .AddAWSEC2Detector();  
  
var tracerProvider = Sdk.CreateTracerProviderBuilder()  
    .AddSource(serviceName)  
    .ConfigureResource(resource =>  
        resource  
            .AddAWSEC2Detector()  
            .AddService(  
                serviceName: serviceName,  
                serviceVersion: serviceVersion  
            )  
        )  
    .AddOtlpExporter() // default address localhost:4317  
    .SetSampler(new TraceIdRatioBasedSampler(1.00))  
    .Build();  
  
Sdk.SetDefaultTextMapPropagator(new AWSXRayPropagator()); // configure X-Ray  
propagator
```

Criar dados de rastreamento manualmente

With X-Ray SDK

Com o SDK do X-Ray, os métodos `BeginSegment` e `BeginSubsegment` eram necessários para criar manualmente segmentos e subsegmentos do X-Ray.

```
using Amazon.XRay.Recorder.Core;

AWSXRayRecorder.Instance.BeginSegment("segment name"); // generates `TraceId` for
you
try
{
    // Do something here
    // can create custom subsegments
    AWSXRayRecorder.Instance.BeginSubsegment("subsegment name");
    try
    {
        DoSometing();
    }
    catch (Exception e)
    {
        AWSXRayRecorder.Instance.AddException(e);
    }
    finally
    {
        AWSXRayRecorder.Instance.EndSubsegment();
    }
}
catch (Exception e)
{
    AWSXRayRecorder.Instance.AddException(e);
}
finally
{
    AWSXRayRecorder.Instance.EndSegment();
}
```

With OpenTelemetry SDK

Em .NET, você pode usar a API de atividades para criar extensões personalizadas para monitorar o desempenho de atividades internas que não são capturadas pelas bibliotecas de instrumentação. Somente extensões do tipo servidor são convertidas em segmentos do X-Ray. Todas as outras extensões são convertidas em subsegmentos do X-Ray.

Você pode criar quantas instâncias `ActivitySource` forem necessárias, mas é recomendável ter apenas uma para uma aplicação/serviço inteiro.

```
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
    }
}
```

Adicionar anotações e metadados aos rastreamentos com o SDK OpenTelemetry

Você também pode adicionar pares de valor-chave personalizados como atributos às extensões usando o método `SetTag` em uma atividade. Por padrão, todos os atributos de extensão serão convertidos em metadados nos dados brutos do X-Ray. Para garantir que um atributo seja convertido em uma anotação, e não em metadados, adicione a chave desses atributos à lista do atributo `aws.xray.annotations`.

```
using (var activity = activitySource.StartActivity("ActivityName",
    ActivityKind.Server)) // this will be translated to a X-Ray Segment
{
    activity.SetTag("metadataKey", "metadataValue");
    activity.SetTag("annotationKey", "annotationValue");
    string[] annotationKeys = {"annotationKey"};
    activity.SetTag("aws.xray.annotations", annotationKeys);

    // Do something here

    using (var internalActivity = activitySource.StartActivity("ActivityName",
        ActivityKind.Internal)) // this will be translated to an X-Ray Subsegment
    {
        // Do something here
    }
}
```

Com instrumentação OpenTelemetry automática

Se você estiver usando uma solução de instrumentação OpenTelemetry automática para .NET e precisar executar instrumentação manual em seu aplicativo, por exemplo, para instrumentar o código dentro do próprio aplicativo para seções que não são cobertas por nenhuma biblioteca de instrumentação automática.

Como só pode haver um `TracerProvider` global, a instrumentação manual não deve instanciar seu próprio `TracerProvider` se usada em conjunto com a instrumentação automática. Quando `TracerProvider` usado, o rastreamento manual personalizado funciona da mesma forma ao usar instrumentação automática ou instrumentação manual por meio do SDK. OpenTelemetry

Rastrear solicitações recebidas (instrumentação do ASP.NET e do ASP.NET Core)

With X-Ray SDK

Para instrumentar solicitações atendidas pela aplicação ASP.NET, consulte <https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-dotnet-messagehandler.html> para obter informações sobre como chamar o `RegisterXRay` no método `Init` do seu arquivo `global.asax`.


```
AWSXRayAspNet.RegisterXRay(this, "MyApp");
```

Para instrumentar as solicitações atendidas pela sua aplicação do ASP.NET Core, o método `UseXRay` é chamado antes de qualquer outro middleware no método `Configure` da sua classe de inicialização.

```
app.UseXRay("MyApp");
```

With OpenTelemetry SDK

OpenTelemetry também fornece bibliotecas de instrumentação para coletar rastreamentos para solicitações da Web recebidas para ASP.NET e ASP.NET core. A seção a seguir lista as etapas necessárias para adicionar e habilitar essas instrumentações de biblioteca para sua OpenTelemetry configuração, incluindo como adicionar instrumentação [ASP.NET ou ASP.NET core](#) ao criar o Tracer Provider.

Para obter informações sobre como habilitar `OpenTelemetry .Instrumentation. AspNet`, consulte [Etapas para habilitar OpenTelemetry .Instrumentation. AspNet](#) para obter informações sobre como habilitar `OpenTelemetry .Instrumentation. AspNetCore`, consulte [Etapas para habilitar OpenTelemetry .Instrumentation. AspNetCore](#).

AWS Instrumentação do SDK

With X-Ray SDK

Instale todos os clientes do AWS SDK `RegisterXRayForAllServices()` ligando.

```
using Amazon.XRay.Recorder.Handlers.AwsSdk;  
AWS SDKHandler.RegisterXRayForAllServices(); //place this before any instantiation of  
AmazonServiceClient  
AmazonDynamoDBClient client = new AmazonDynamoDBClient(RegionEndpoint.USWest2); //  
AmazonDynamoDBClient is automatically registered with X-Ray
```

Use um dos métodos a seguir para instrumentação específica do cliente de AWS serviço.

```
AWSSDKHandler.RegisterXRay<IAmazonDynamoDB>(); // Registers specific type of
AmazonServiceClient : All instances of IAmazonDynamoDB created after this line are
registered
AWSSDKHandler.RegisterXRayManifest(String path); // To configure custom AWS Service
Manifest file. This is optional, if you have followed "Configuration" section
```

With OpenTelemetry SDK

Para o exemplo de código a seguir, você precisará da seguinte dependência:

```
dotnet add package OpenTelemetry.Instrumentation.AWS
```

Para instrumentar o AWS SDK, atualize a configuração do OpenTelemetry SDK onde o Global TracerProvider está configurado.

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
        .AddAWSInstrumentation()
        ...
```

Instrumentar chamadas HTTP de saída

With X-Ray SDK

O SDK .NET do X-Ray rastreia as chamadas HTTP de saída por meio dos métodos de extensão `GetResponseTraced()` ou `GetAsyncResponseTraced()` durante o uso de `System.Net.HttpWebRequest` ou usando o manipulador `HttpClientXRayTracingHandler` durante o uso de `System.Net.Http.HttpClient`.

With OpenTelemetry SDK

Para o exemplo de código a seguir, você precisará da seguinte dependência:

```
dotnet add package OpenTelemetry.Instrumentation.Http
```

Para instrumentar `System.Net.Http.HttpClient` e `System.Net.HttpWebRequest`, atualize a configuração do OpenTelemetry SDK onde o Global TracerProvider está configurado.

```
builder.Services.AddOpenTelemetry()  
    ...  
    .WithTracing(tracing => tracing  
        .AddHttpClientInstrumentation()  
        ...
```

Suporte de instrumentação para outras bibliotecas

Você pode pesquisar e filtrar o OpenTelemetry Registro para bibliotecas de instrumentação.NET para descobrir se OpenTelemetry oferece suporte à instrumentação para sua biblioteca. Consulte o [Registry](#) para começar a pesquisar.

Instrumentação do Lambda

With X-Ray SDK

O procedimento a seguir foi necessário para usar o SDK do X-Ray com o Lambda:

1. Habilitar o Rastreamento ativo na sua função do Lambda
2. O serviço Lambda cria um segmento que representa a invocação do seu manipulador
3. Criar subsegmentos ou bibliotecas de instrumentos usando o SDK do X-Ray

With OpenTelemetry-based solutions

Você pode instrumentar automaticamente seu Lambda com camadas personalizadas do AWS Lambda. Existem duas soluções:

- Camada [lambda \(recomendada\) CloudWatch do Application Signals](#)
- Para um melhor desempenho, talvez você queira considerar o uso `OpenTelemetry Manual Instrumentation` para gerar OpenTelemetry rastreamentos para sua função Lambda.

OpenTelemetry instrumentação manual para Lambda AWS

Veja a seguir um exemplo do código da função do Lambda (sem instrumentação).

```
using System;
using System.Text;
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using Amazon.S3.Model;

// Assembly attribute to enable Lambda function logging
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace ExampleLambda;

public class ListBucketsHandler
{
    private static readonly AmazonS3Client s3Client = new();

    // new Lambda function handler passed in
    public async Task<string> HandleRequest(object input, ILambdaContext context)
    {
        try
        {
            var DoListBucketsAsyncResponse = await DoListBucketsAsync();
            context.Logger.LogInformation($"Results:
{DoListBucketsAsyncResponse.Buckets}");

            context.Logger.LogInformation($"Successfully called ListBucketsAsync");
            return "Success!";
        }
        catch (Exception ex)
        {
            context.Logger.LogError($"Failed to call ListBucketsAsync: {ex.Message}");
            throw;
        }
    }

    private async Task<ListBucketsResponse> DoListBucketsAsync()
    {
        try
```

```
{
    var putRequest = new ListBucketsRequest
    {
    };

    var response = await s3Client.ListBucketsAsync(putRequest);
    return response;
}
catch (AmazonS3Exception ex)
{
    throw new Exception($"Failed to call ListBucketsAsync: {ex.Message}", ex);
}
}
```

Para instrumentar manualmente seu manipulador do Lambda e o cliente do Amazon S3, faça o seguinte:

1. Instanciar um `TracerProvider` — Recomenda-se que `TracerProvider` seja configurado com um `XrayUdpSpanExporter`, um `ParentBased Always On Sampler` e um `Resource` com `service.name` definido como o nome da função Lambda.
2. Instrumente o cliente Amazon S3 com a instrumentação do OpenTelemetry AWS SDK ligando para adicionar a instrumentação do cliente do SDK `AddAWSInstrumentation()` ao `AWS TracerProvider`
3. Crie uma função wrapper com a mesma assinatura da função do Lambda original. Chame a API `AWSLambdaWrapper.Trace()` e passe `TracerProvider`, a função do Lambda original, e suas entradas como parâmetros. Defina a função wrapper como a entrada do manipulador do Lambda.

Para o exemplo de código a seguir, você precisará das seguintes dependências:

```
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
```

O código a seguir demonstra a função do Lambda após as alterações necessárias. Você pode criar extensões personalizadas adicionais para complementar as extensões fornecidas automaticamente.

```

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);
    }
}
}

```

Ao invocar esse Lambda, você verá o seguinte rastreamento no Trace Map no console: CloudWatch



Migrar para Python OpenTelemetry

Este guia ajuda você a migrar aplicativos Python do X-Ray SDK OpenTelemetry para a instrumentação. Ele abrange abordagens de instrumentação automática e manual, com exemplos de código para cenários comuns.

Seções

- [Soluções de instrumentação automática de código zero](#)
- [Instrumente suas aplicações manualmente](#)
- [Inicialização da configuração de rastreamento](#)
- [Rastreamento de solicitações de entrada](#)
- [AWS Instrumentação do SDK](#)
- [Instrumentar chamadas HTTP de saída por meio de solicitações](#)
- [Suporte de instrumentação para outras bibliotecas](#)
- [Criar dados de rastreamento manualmente](#)
- [Instrumentação do Lambda](#)

Soluções de instrumentação automática de código zero

Com o X-Ray SDK, você precisava modificar o código do aplicativo para rastrear solicitações. OpenTelemetry oferece soluções de instrumentação automática de código zero para rastrear solicitações. Com OpenTelemetry, você tem a opção de usar soluções de instrumentação automática de código zero para rastrear solicitações.

Código zero com OpenTelemetry instrumentações automáticas baseadas

1. [Usando a AWS Distro for OpenTelemetry \(ADOT\) Auto-instrumentation para Python — Para instrumentação automática para aplicativos Python, consulte Rastreamento e métricas com a Distro for Python Auto-Instrumentation. AWS OpenTelemetry](#)

(Opcional) Você também pode ativar o CloudWatch Application Signals ao instrumentar automaticamente seus aplicativos AWS com a instrumentação automática ADOT Python para monitorar a integridade atual do aplicativo e acompanhar o desempenho de longo prazo do aplicativo em relação aos seus objetivos de negócios. O Application Signals fornece uma visualização unificada e centrada em aplicativos para aplicativos, serviços e dependências, além de ajudar você a monitorar e realizar a triagem da integridade do aplicativo.

2. [Usando a instrumentação automática de código zero do OpenTelemetry Python — Para instrumentação automática com o Python, consulte Instrumentação de código zero do Python. OpenTelemetry](#)

Instrumente suas aplicações manualmente

Você pode instrumentar suas aplicações manualmente com o comando `pip`.

With X-Ray SDK

```
pip install aws-xray-sdk
```

With OpenTelemetry SDK

```
pip install opentelemetry-api opentelemetry-sdk opentelemetry-exporter-otlp  
opentelemetry-propagator-aws-xray
```

Inicialização da configuração de rastreamento

With X-Ray SDK

No X-Ray, o `xray_recorder` global é inicializado e usado para gerar segmentos e subsegmentos.

With OpenTelemetry SDK

Note

No momento, o X-Ray Remote Sampling não está disponível para ser configurado para OpenTelemetry Python. No entanto, o suporte para a amostragem remota do X-Ray está atualmente disponível por meio da instrumentação automática do ADOT para Python.

Em OpenTelemetry, você precisa inicializar um `globalTracerProvider`. Ao usar este `TracerProvider`, você pode adquirir um [Rastreador](#) que pode ser usado para gerar extensões em qualquer lugar da sua aplicação. É recomendável configurar os seguintes componentes:

- `OTLPSpanExporter`— Necessário para exportar traços para o CloudWatch OpenTelemetry Agente/Coletor
- Um propagador de AWS raio-X — necessário para propagar o contexto de rastreamento para AWS serviços integrados ao X-Ray

```
from opentelemetry import (
    trace,
    propagate
)
from opentelemetry.sdk.trace import TracerProvider

from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.exporter.otlp.proto.http.trace_exporter import OTLPSpanExporter
from opentelemetry.sdk.trace.export import BatchSpanProcessor
from opentelemetry.propagators.aws import AwsXRayPropagator

# Sends generated traces in the OTLP format to an OTel Collector running on port
# 4318
otlp_exporter = OTLPSpanExporter(endpoint="http://localhost:4318/v1/traces")
# Processes traces in batches as opposed to immediately one after the other
span_processor = BatchSpanProcessor(otlp_exporter)
# More configurations can be done here. We will visit them later.

# Sets the global default tracer provider
provider = TracerProvider(active_span_processor=span_processor)
trace.set_tracer_provider(provider)

# Configures the global propagator to use the X-Ray Propagator
propagate.set_global_textmap(AwsXRayPropagator())

# Creates a tracer from the global tracer provider
tracer = trace.get_tracer("my.tracer.name")
# Use this tracer to create Spans
```

Com instrumentação automática do ADOT para Python

Você pode usar a instrumentação automática ADOT para Python para configurar OpenTelemetry automaticamente seus aplicativos Python. Ao usar a instrumentação automática ADOT, você não precisa fazer alterações manuais no código para rastrear solicitações recebidas ou rastrear bibliotecas, como os AWS clientes SDK ou HTTP. Para obter mais informações, consulte [Rastreamento e métricas com a AWS distro for Python OpenTelemetry](#) Auto-Instrumentation.

A instrumentação automática do ADOT para Python é compatível com:

- Amostragem remota do X-Ray por meio de variável de ambiente `export OTEL_TRACES_SAMPLER=xray`
- Propagação do contexto de rastreamento do X-Ray (habilitada por padrão)
- Detecção de recursos (a detecção de recursos para ambientes Amazon EC2, Amazon ECS e Amazon EKS é ativada por padrão)
- As instrumentações automáticas da biblioteca para todas as OpenTelemetry instrumentações suportadas estão habilitadas por padrão. Você pode desativá-las seletivamente por meio da variável de ambiente `OTEL_PYTHON_DISABLED_INSTRUMENTATIONS` (todas são habilitadas por padrão)
- Criação manual de expansões

De plug-ins de serviço X-Ray a provedores de OpenTelemetry AWS recursos

O X-Ray SDK fornece plug-ins que você pode adicionar ao `xray_recorder` para capturar as informações específicas da plataforma do serviço hospedado, como Amazon, EC2 Amazon ECS e Elastic Beanstalk. É semelhante aos provedores de recursos, pois captura OpenTelemetry as informações como atributos de recursos. Há vários provedores de recursos disponíveis para diferentes AWS plataformas.

- Comece instalando o pacote AWS de extensão, `pip install opentelemetry-sdk-extension-aws`
- Configure o detector de recursos desejado. O exemplo a seguir mostra como configurar o provedor de EC2 recursos da Amazon no OpenTelemetry SDK

```
from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
```

```
from opentelemetry.sdk.extension.aws.resource.ec2 import (
    AwsEc2ResourceDetector,
)
from opentelemetry.sdk.resources import get_aggregated_resources

provider = TracerProvider(
    active_span_processor=span_processor,
    resource=get_aggregated_resources([
        AwsEc2ResourceDetector(),
    ])
)

trace.set_tracer_provider(provider)
```

Rastreamento de solicitações de entrada

With X-Ray SDK

O SDK Python do X-Ray é compatível com estruturas de aplicações como Django, Flask e Bottle no rastreamento de solicitações recebidas de aplicações Python executadas nelas. Para fazer isso, adicione `XRayMiddleware` à aplicação de cada estrutura.

With OpenTelemetry SDK

OpenTelemetry fornece instrumentações para [Django](#) e [Flask](#) por meio de bibliotecas de instrumentação específicas. [Não há instrumentação para Bottle disponível em OpenTelemetry, as aplicações ainda podem ser rastreadas usando a instrumentação WSGI. OpenTelemetry](#)

Para o exemplo de código a seguir, você precisa da seguinte dependência:

```
pip install opentelemetry-instrumentation-flask
```

Você deve inicializar o OpenTelemetry SDK e registrar o global `TracerProvider` antes de adicionar instrumentações à estrutura do seu aplicativo. Sem isso, as operações de rastreamento serão no-ops. Após configurar o `TracerProvider` global, você pode usar o instrumentor para sua estrutura de aplicações. O exemplo a seguir demonstra um aplicativo Flask.

```
from flask import Flask
```

```
from opentelemetry import trace
from opentelemetry.instrumentation.flask import FlaskInstrumentor
from opentelemetry.sdk.extension.aws.resource import AwsEc2ResourceDetector
from opentelemetry.sdk.resources import get_aggregated_resources
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.trace.export import BatchSpanProcessor, ConsoleSpanExporter

provider = TracerProvider(resource=get_aggregated_resources(
    [
        AwsEc2ResourceDetector(),
    ]))

processor = BatchSpanProcessor(ConsoleSpanExporter())
provider.add_span_processor(processor)
trace.set_tracer_provider(provider)

# Creates a tracer from the global tracer provider
tracer = trace.get_tracer("my.tracer.name")

app = Flask(__name__)

# Instrument the Flask app
FlaskInstrumentor().instrument_app(app)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

AWS Instrumentação do SDK

With X-Ray SDK

O X-Ray Python SDK rastreia a solicitação do cliente AWS SDK aplicando patches na biblioteca. `botocore` Para obter mais informações, consulte [Rastreamento de chamadas do AWS SDK com o X-Ray SDK para Python](#). Na sua aplicação, o método `patch_all()` é usado para instrumentar todas as bibliotecas ou corrigir seletivamente usando as bibliotecas `botocore` ou `boto3` com

`patch((['botocore']))`. Qualquer um dos métodos escolhidos instrumenta todos os clientes do Boto3 na sua aplicação e gera um subsegmento para qualquer chamada feita usando esses clientes.

With OpenTelemetry SDK

Para o exemplo de código a seguir, você precisará da seguinte dependência:

```
pip install opentelemetry-instrumentation-botocore
```

Use a instrumentação [OpenTelemetry Botocore programaticamente para instrumentar](#) todos os clientes do Boto3 em seu aplicativo. Os exemplos a seguir demonstram a instrumentação do 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}")
```

Instrumentar chamadas HTTP de saída por meio de solicitações

With X-Ray SDK

O X-Ray Python SDK rastreia chamadas HTTP de saída por meio de solicitações corrigindo a biblioteca de solicitações. Para obter mais informações, consulte [Rastrear chamadas para serviços web HTTP downstream usando o SDK do X-Ray para Python](#). Na sua aplicação, você pode usar o método `patch_all()` para instrumentar todas as bibliotecas ou corrigir seletivamente as bibliotecas de solicitações usando `patch(['requests'])`. Qualquer uma das opções instrumenta a biblioteca `requests`, gerando um subsegmento para qualquer chamada feita por meio de `requests`.

With OpenTelemetry SDK

Para o exemplo de código a seguir, você precisará da seguinte dependência:

```
pip install opentelemetry-instrumentation-requests
```

Use a Instrumentação de OpenTelemetry Solicitações programaticamente para instrumentar a biblioteca de solicitações para gerar rastreamentos para solicitações HTTP feitas por ela em seu

aplicativo. Para obter mais informações, consulte [OpenTelemetry solicitações de instrumentação](#). O exemplo a seguir demonstra a instrumentação da biblioteca `requests`.

```
from opentelemetry.instrumentation.requests import RequestsInstrumentor

# Instrument Requests
RequestsInstrumentor().instrument()

...

example_session = requests.Session()
example_session.get(url="https://example.com")
```

Como alternativa, você também pode instrumentar a biblioteca `urllib3` subjacente para rastrear solicitações HTTP:

```
# pip install opentelemetry-instrumentation-urllib3
from opentelemetry.instrumentation.urllib3 import URLLib3Instrumentor

# Instrument urllib3
URLLib3Instrumentor().instrument()

...

example_session = requests.Session()
example_session.get(url="https://example.com")
```

Suporte de instrumentação para outras bibliotecas

Você pode encontrar a lista completa de instrumentações de biblioteca compatíveis com OpenTelemetry Python em [Bibliotecas, estruturas, servidores](#) de aplicativos e JVMs

Como alternativa, você pode pesquisar no OpenTelemetry Registro para descobrir se OpenTelemetry suporta instrumentação. Consulte o [Registry](#) para começar a pesquisar.

Criar dados de rastreamento manualmente

Você pode criar segmentos e subsegmentos usando o `xray_recorder` na sua aplicação Python. Para obter mais informações, consulte [Instrumentar o código Python manualmente](#). Você também pode adicionar manualmente anotações e metadados aos dados de rastreamento.

Criação de extensões com OpenTelemetry o SDK

Use a `start_as_current_span` API para iniciar uma extensão e configurá-la para criar extensões. Para ver exemplos de criação de extensões, consulte [Criar extensões](#). Depois que um período é iniciado e está no escopo atual, você pode adicionar mais informações a ele adicionando atributos, eventos, exceções, links e assim por diante. Da mesma forma que temos segmentos e subsegmentos no X-Ray, existem diferentes tipos de extensões. OpenTelemetry Somente os intervalos do tipo `SERVER` são convertidos em segmentos do X-Ray, enquanto outros são convertidos em subsegmentos do X-Ray.

```
from opentelemetry import trace
from opentelemetry.trace import SpanKind

import time

tracer = trace.get_tracer("my.tracer.name")

# Create a new span to track some work
with tracer.start_as_current_span("parent", kind=SpanKind.SERVER) as parent_span:
    time.sleep(1)

    # Create a nested span to track nested work
    with tracer.start_as_current_span("child", kind=SpanKind.CLIENT) as child_span:
        time.sleep(2)
        # the nested span is closed when it's out of scope

    # Now the parent span is the current span again
    time.sleep(1)

    # This span is also closed when it goes out of scope
```

Adicionar anotações e metadados aos rastreamentos com o SDK OpenTelemetry

O X-Ray Python SDK fornece anotações `put_annotation` e `put_metadata` metadados separados APIs a um rastreamento. No OpenTelemetry SDK, as anotações e os metadados são simplesmente atributos em um intervalo, adicionados por meio da API. `set_attribute`

Os atributos de extensão, que você quer que sejam anotações em um rastreamento, são adicionados à chave reservada `aws.xray.annotations` cujo valor é uma lista de pares de anotações de valor-chave. Todos os outros atributos de extensão se tornam metadados no segmento ou no subsegmento convertido.

Além disso, se você estiver usando o coletor ADOT, poderá configurar quais atributos de extensão deverão ser convertidos em anotações do X-Ray especificando o `indexed_attributes` na configuração do coletor.

O exemplo abaixo demonstra como adicionar anotações e metadados a um rastreamento usando o SDK. 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")
```

Instrumentação do Lambda

Para monitorar suas funções do lambda no X-Ray, você habilita o X-Ray e adiciona as permissões apropriadas ao perfil de invocação da função. Além disso, se você rastreia solicitações downstream da sua função, está instrumentando o código com o X-Ray Python SDK.

Com OpenTelemetry o for X-Ray, é recomendável usar a camada lambda do CloudWatch Application Signals com o [Application Signals](#) desativado. Isso instrumenta automaticamente sua função e gera

extensões para a invocação da função e qualquer solicitação downstream da sua função. Além do rastreamento, se você tiver interesse em usar o Application Signals para monitorar a integridade da sua função, consulte [Ativar suas aplicações no Lambda](#).

- Encontre o ARN da camada Lambda necessário para sua função em [AWS Lambda](#) Layer for e adicione-o. OpenTelemetry ARNs
- Defina as seguintes variáveis de ambiente para sua função.
 - `AWS_LAMBDA_EXEC_WRAPPER=/opt/otel-instrument`: carrega a instrumentação automática para a função
 - `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`: desabilitará o monitoramento do Application Signals

Criar manualmente extensões com instrumentação do Lambda

Além disso, você pode gerar extensões personalizadas na sua função para monitorar o trabalho. Você pode fazer isso usando somente o pacote `opentelemetry-api` em conjunto com a instrumentação automática da camada do Lambda do Application Signals.

1. Inclua a `opentelemetry-api` como uma dependência na sua função
2. O snippet de código a seguir é um exemplo para gerar extensões personalizadas

```
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
    }
```

Migrar para Ruby OpenTelemetry

Para migrar seus aplicativos Ruby do X-Ray SDK para a OpenTelemetry instrumentação, use os seguintes exemplos de código e orientações para instrumentação manual.

Seções

- [Instrumente manualmente suas soluções com o SDK](#)
- [Rastrear solicitações recebidas \(instrumentação Rails\)](#)
- [AWS Instrumentação do SDK](#)
- [Instrumentar chamadas HTTP de saída](#)
- [Suporte de instrumentação para outras bibliotecas](#)
- [Criar dados de rastreamento manualmente](#)
- [Instrumentação manual Lambda](#)

Instrumente manualmente suas soluções com o SDK

Tracing setup with X-Ray SDK

O SDK do X-Ray para Ruby exigia que você configurasse seu código com plug-ins de serviço.

```
require 'aws-xray-sdk'

XRay.recorder.configure(plugins: [:ec2, :elastic_beanstalk])
```

Tracing setup with OpenTelemetry SDK

Note

No momento, a amostragem remota X-Ray não está disponível para ser configurada para OpenTelemetry Ruby.

Para um aplicativo Ruby on Rails, coloque seu código de configuração em um inicializador Rails. Para obter mais informações, consulte [Conceitos básicos do](#) . Para todos os programas Ruby instrumentados manualmente, você deve usar o `OpenTelemetry::SDK.configure` método para configurar o SDK do OpenTelemetry Ruby.

Primeiro, instale os seguintes pacotes:

```
bundle add opentelemetry-sdk opentelemetry-exporter-otlp opentelemetry-propagator-xray
```

Em seguida, configure o OpenTelemetry SDK por meio do código de configuração que é executado quando o programa é inicializado. É recomendável configurar os seguintes componentes:

- **OTLP Exporter**— Necessário para exportar traços para o CloudWatch agente e coletor OpenTelemetry
- **An AWS X-Ray Propagator**— Necessário para propagar o contexto de rastreamento para AWS serviços integrados ao X-Ray

```
require 'opentelemetry-sdk'
require 'opentelemetry-exporter-otlp'

# Import the gem containing the AWS X-Ray for OTel Ruby ID Generator and propagator
require 'opentelemetry-propagator-xray'

OpenTelemetry::SDK.configure do |c|
  c.service_name = 'my-service-name'

  c.add_span_processor(
    # Use the BatchSpanProcessor to send traces in groups instead of one at a time
    OpenTelemetry::SDK::Trace::Export::BatchSpanProcessor.new(
      # Use the default OTLP 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"
      )
    )
  )
end
```

```
# The X-Ray Propagator injects the X-Ray Tracing Header into downstream calls
c.propagators = [OpenTelemetry::Propagator::XRay::TextMapPropagator.new]
end
```

OpenTelemetry SDKs também têm o conceito de instrumentações de biblioteca. Habilitá-los criará automaticamente extensões para bibliotecas como o AWS SDK. OpenTelemetry fornece a opção de ativar todas as instrumentações da biblioteca ou especificar quais instrumentações da biblioteca devem ser ativadas.

Para habilitar todas as instrumentações, primeiro instale o pacote `opentelemetry-instrumentation-all`:

```
bundle add opentelemetry-instrumentation-all
```

Em seguida, atualize a configuração para habilitar todas as instrumentações de biblioteca, conforme mostrado abaixo:

```
require 'opentelemetry/instrumentation/all'
...

OpenTelemetry::SDK.configure do |c|
  ...

  c.use_all() # Enable all instrumentations
end
```

OpenTelemetry SDKs também têm o conceito de instrumentações de biblioteca. Habilitá-los criará automaticamente extensões para bibliotecas como o AWS SDK. OpenTelemetry fornece a opção de ativar todas as instrumentações da biblioteca ou especificar quais instrumentações da biblioteca devem ser ativadas.

Para habilitar todas as instrumentações, primeiro instale o pacote `opentelemetry-instrumentation-all`:

```
bundle add opentelemetry-instrumentation-all
```

Em seguida, atualize a configuração para habilitar todas as instrumentações de biblioteca, conforme mostrado abaixo:

```
require 'opentelemetry/instrumentation/all'
...

OpenTelemetry::SDK.configure do |c|
  ...

  c.use_all() # Enable all instrumentations
end
```

Rastrear solicitações recebidas (instrumentação Rails)

With X-Ray SDK

Com o SDK do X-Ray, o rastreamento do X-Ray é configurado para a estrutura do Rails na inicialização.

Exemplo — `config/initializers/aws_xray.rb`

```
Rails.application.config.xray = {
  name: 'my app',
  patch: %I[net_http aws_sdk],
  active_record: true
}
```

With OpenTelemetry SDK

Primeiro, instale os seguintes pacotes:

```
bundle add opentelemetry-instrumentation-rack opentelemetry-instrumentation-
rails opentelemetry-instrumentation-action_pack opentelemetry-instrumentation-
active_record opentelemetry-instrumentation-action_view
```

Em seguida, atualize a configuração para habilitar a instrumentação para sua aplicação Rails, conforme mostrado abaixo:

```
# 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 Instrumentação do SDK

With X-Ray SDK

Para instrumentar as AWS solicitações de saída do AWS SDK, os clientes do AWS SDK são corrigidos com X-Ray, como no exemplo a seguir:

```
require 'aws-xray-sdk'
require 'aws-sdk-s3'

# Patch AWS SDK clients
XRay.recorder.configure(plugings: [:aws_sdk])

# Use the instrumented client
s3 = Aws::S3::Client.new
s3.list_buckets
```

With OpenTelemetry SDK

AWS O SDK for Ruby V3 fornece suporte para gravação e emissão de traços. OpenTelemetry Para obter informações sobre como configurar um cliente de serviço, consulte [Configuração de recursos de observabilidade no AWS SDK OpenTelemetry for Ruby](#).

Instrumentar chamadas HTTP de saída

Ao fazer chamadas HTTP para serviços externos, talvez seja necessário instrumentar manualmente as chamadas se a instrumentação automática não estiver disponível ou não fornecer detalhes suficientes.

With X-Ray SDK

Para instrumentar chamadas downstream, o SDK do X-Ray para Ruby foi usado para aplicar patches à biblioteca `net/http` usada pela aplicação:

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http]
}

XRay.recorder.configure(config)
```

With OpenTelemetry SDK

Para habilitar o uso da `net/http` instrumentação OpenTelemetry, primeiro instale o `opentelemetry-instrumentation-net_http` pacote:

```
bundle add opentelemetry-instrumentation-net_http
```

Em seguida, atualize a configuração para habilitar a instrumentação de `net/http`, conforme mostrado abaixo:

```
OpenTelemetry::SDK.configure do |c|
  ...

  c.use 'OpenTelemetry::Instrumentation::Net::HTTP'
  ...
end
```

Suporte de instrumentação para outras bibliotecas

Você pode encontrar a lista completa de instrumentações de biblioteca suportadas para OpenTelemetry Ruby em: [opentelemetry-ruby-contrib](#)

Como alternativa, você pode pesquisar no OpenTelemetry Registro para descobrir se OpenTelemetry suporta instrumentação. Para obter mais informações, consulte [Registry](#).

Criar dados de rastreamento manualmente

With X-Ray SDK

Ao usar o X-Ray, o pacote `aws-xray-sdk` necessário para criar manualmente segmentos e seus subsegmentos filho para rastrear sua aplicação. Você também pode ter adicionado anotações e metadados do X-Ray aos seus segmentos ou subsegmentos:

```
require 'aws-xray-sdk'
...

# Start a segment
segment = XRay.recorder.begin_segment('my-service')

# Add annotations (indexed key-value pairs)
segment.annotations[:user_id] = 'user-123'
segment.annotations[:payment_status] = 'completed'

# Add metadata (non-indexed data)
segment.metadata[:order] = {
  id: 'order-456',
  items: [
    { product_id: 'prod-1', quantity: 2 },
    { product_id: 'prod-2', quantity: 1 }
  ],
  total: 67.99
}

# Add metadata to a specific namespace
segment.metadata(namespace: 'payment') do |metadata|
  metadata[:transaction_id] = 'tx-789'
  metadata[:payment_method] = 'credit_card'
end
```

```
# Create a subsegment with annotations and metadata
segment.subsegment('payment-processing') do |subsegment1|
  subsegment1.annotations[:payment_id] = 'pay-123'
  subsegment1.metadata[:details] = { amount: 67.99, currency: 'USD' }

  # Create a nested subsegment
  subsegment1.subsegment('operation-2') do |subsegment2|
    # Do more work...
  end
end

# Close the segment
segment.close
```

With OpenTelemetry SDK

Você pode usar extensões personalizadas para monitorar o desempenho de atividades internas que não são capturadas pelas bibliotecas de instrumentação. Somente extensões do tipo Servidor são convertidas em segmentos do X-Ray. Todas as outras extensões são convertidas em subsegmentos do X-Ray. Por padrão, as extensões são INTERNAL.

Primeiro, crie um Rastreador para gerar extensões, que podem ser obtidas por meio do método `OpenTelemetry.tracer_provider.tracer('<YOUR_TRACER_NAME>')`. Isso fornecerá uma instância do Tracer registrada globalmente na OpenTelemetry configuração do seu aplicativo. É comum ter um único rastreador para um aplicativo inteiro. Crie um OpenTelemetry rastreador e use-o para criar extensões:

```
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
```

Adicionar anotações e metadados aos rastreamentos com o SDK OpenTelemetry

Use o método `set_attribute` para adicionar atributos a cada extensão. Por padrão, todos esses atributos de extensão serão convertidos em metadados nos dados brutos do X-Ray. Para garantir que um atributo seja convertido em uma anotação, e não em metadados, adicione a chave desses atributos à lista do atributo `aws.xray.annotations`. Para obter mais informações, consulte [Enable The Customized X-Ray Annotations](#).

```
# 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
```

Instrumentação manual Lambda

With X-Ray SDK

Depois que o Rastreamento ativo for habilitado no Lambda, não haverá necessidade de configurações adicionais para usar o SDK do X-Ray. O Lambda criará um segmento representando a invocação do manipulador do Lambda, e você poderá criar subsegmentos ou bibliotecas de instrumentos usando o SDK do X-Ray sem nenhuma configuração adicional.

With OpenTelemetry SDK

Considere o seguinte exemplo de código de função do Lambda (sem instrumentação):

```
require 'json'
def lambda_handler(event:, context:)
  # TODO implement
  { statusCode: 200, body: JSON.generate('Hello from Lambda!') }
end
```

Para instrumentar manualmente seu Lambda, é necessário:

1. Adicionar os seguintes gems para seu Lambda

```
gem 'opentelemetry-sdk'
gem 'opentelemetry-exporter-otlp'
gem 'opentelemetry-propagator-xray'
gem 'aws-distro-opentelemetry-exporter-xray-udp'
gem 'opentelemetry-instrumentation-aws_lambda'
gem 'opentelemetry-propagator-xray', '~> 0.24.0' # Requires version v0.24.0 or higher
```

2. Inicialize o OpenTelemetry SDK fora do seu Lambda Handler. Recomenda-se que o OpenTelemetry SDK seja configurado com:
 1. Um processador de extensões simples com um exportador de extensões do UDP do X-Ray para enviar rastreamentos ao endpoint UDP X-Ray do Lambda
 2. Um propagador do X-Ray Lambda
 3. Configuração de `service_name` a ser definida para o nome da função do Lambda
3. Na sua classe de manipulador Lambda, adicione as seguintes linhas para instrumentar seu manipulador Lambda:

```
class Handler
  extend OpenTelemetry::Instrumentation::AwsLambda::Wrap
  ...

  instrument_handler :process
end
```

O código a seguir demonstra a função do Lambda após as alterações necessárias. Você pode criar extensões personalizadas adicionais para complementar as extensões fornecidas automaticamente.

```
require 'json'
require 'opentelemetry-sdk'
require 'aws/distro/opentelemetry/exporter/xray/udp'
require 'opentelemetry/propagator/xray'
require 'opentelemetry/instrumentation/aws_lambda'

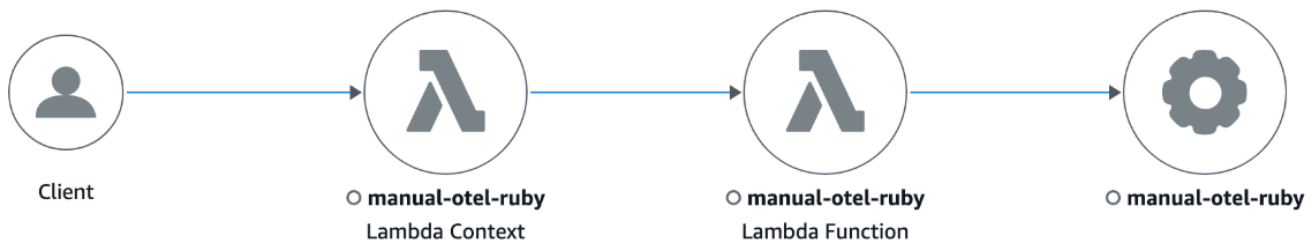
# Initialize OpenTelemetry SDK outside handler
OpenTelemetry::SDK.configure do |c|
  # Configure the AWS Distro for OpenTelemetry X-Ray Lambda exporter
  c.add_span_processor(
    OpenTelemetry::SDK::Trace::Export::SimpleSpanProcessor.new(
      AWS::Distro::OpenTelemetry::Exporter::XRay::UDP::AWSXRayUDPSpanExporter.new
    )
  )

  # 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
```

Veja a seguir um exemplo de mapa de rastreamento de uma função do Lambda instrumentada escrita no Ruby.



Você também pode usar camadas Lambda OpenTelemetry para configurar seu Lambda. Para obter mais informações, consulte Instrumentação [OpenTelemetry AWS-Lambda](#).

Criar recursos do X-Ray com o AWS CloudFormation

AWS X-Ray está integrado a AWS CloudFormation, um serviço que ajuda a modelar e configurar seus atributos AWS para que você possa gastar menos tempo criando e gerenciando seus atributos e infraestrutura. Você cria um modelo que descreve todos os recursos da AWS desejados, e o CloudFormation provisiona e configura esses recursos para você.

Quando você usa o CloudFormation, você pode reutilizar seu modelo para configurar seus recursos X-Ray repetidamente e de forma consistente. Descreva os recursos uma vez e provisione os mesmos recursos repetidamente em várias regiões e Contas da AWS.

X-Ray e modelos do CloudFormation

Para provisionar e configurar recursos para o X-Ray e serviços relacionados, você deve conhecer os [modelos do CloudFormation](#). Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os atributos que você deseja provisionar nas suas pilhas CloudFormation. Se não estiver familiarizado com o JSON ou o YAML, é possível usar o CloudFormation Designer para ajudar a começar a usar os modelos do CloudFormation. Para mais informações, consulte [O que é o CloudFormation Designer?](#) no Manual do usuário da AWS CloudFormation.

O X-Ray é compatível com a criação dos recursos [AWS::XRay::Group](#), [AWS::XRay::SamplingRule](#), e [AWS::XRay::ResourcePolicy](#) no CloudFormation. Para obter mais informações, incluindo exemplos de modelos JSON e YAML para esses recursos, consulte [X-Ray resource type reference](#) no Guia do usuário do AWS CloudFormation.

Saiba mais sobre o CloudFormation

Para saber mais sobre o CloudFormation, consulte os seguintes recursos:

- [AWS CloudFormation](#)
- [AWS CloudFormation Guia do usuário do](#)
- [CloudFormation Referência de API do](#)
- [AWS CloudFormation Guia do Usuário da Interface de Linha de Comando](#)

Marcar grupos e regras de amostragem do X-Ray

Tags são palavras ou frases que podem ser usadas para identificar e organizar seus recursos da AWS. Você pode adicionar várias tags a cada recurso. Cada tag inclui uma chave e um valor opcional que você define. Por exemplo, uma chave de tag pode ser **domain** e o valor da tag pode ser **example.com**. Você pode pesquisar e filtrar seus recursos de acordo com as tags adicionadas. Para obter mais informações sobre como usar tags, consulte [Marcar recursos da AWS](#) no Guia de referência geral da AWS.

É possível usar tags para aplicar permissões baseadas em tags em distribuições do CloudFront. Para obter mais informações, consulte [Controlar o acesso a recursos da AWS usando tags](#).

Note

No momento, o [Editor de tags](#) e o [AWS Resource Groups](#) não são compatíveis com recursos do X-Ray. Você adiciona e gerencia tags usando a API ou o console do AWS X-Ray.

Você pode aplicar tags a recursos usando o console do X-Ray, a API, a AWS CLI, SDKs e o AWS Tools for Windows PowerShell. Para obter mais informações, consulte a documentação a seguir:

- API do X-Ray: veja as seguintes operações na Referência de API do AWS X-Ray:
 - [ListTagsForResource](#)
 - [CreateSamplingRule](#)
 - [CreateGroup](#)
 - [TagResource](#)
 - [UntagResource](#)
- AWS CLI: consulte [xray](#) na Referência de comandos da AWS CLI.
- SDKs: consulte a documentação do SDK aplicável na página [Documentação da AWS](#)

Note

Se você não conseguir adicionar ou alterar tags em um recurso do X-Ray, ou não conseguir adicionar um recurso que tenha tags específicas, é provável que você não tenha permissões

para realizar essa operação. Para solicitar acesso, entre em contato com um usuário da AWS em sua empresa que tenha permissões de administrador no X-Ray.

Tópicos

- [Restrições de tag](#)
- [Gerenciar tags no console](#)
- [Gerenciar tags na AWS CLI](#)
- [Controlar o acesso a recursos do X-Ray com base em tags](#)

Restrições de tag

As restrições a seguir se aplicam a tags.

- Número máximo de tags por recurso – 50
- Comprimento máximo da chave: 128 caracteres Unicode
- Comprimento máximo do valor: 256 caracteres Unicode
- Valores válidos de chave e valor: a-z, A-Z, 0-9, espaço e os seguintes caracteres: _ . : / = +: e @
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- Não use `aws :` como um prefixo para chaves, pois ele é reservado para uso da AWS.

Note

Não é possível editar ou excluir tags do sistema.

Gerenciar tags no console

Você pode adicionar tags opcionais ao criar uma regra de amostragem ou um grupo do X-Ray. As tags também podem ser alteradas ou excluídas posteriormente no console.

Os procedimentos a seguir explicam como adicionar, editar e excluir tags de grupos e regras de amostragem no console do X-Ray.

Tópicos

- [Adicionar tags a um novo grupo \(console\)](#)
- [Adicionar tags a uma nova regra de amostragem \(console\)](#)
- [Editar ou excluir tags de um grupo \(console\)](#)
- [Editar ou excluir tags de uma regra de amostragem \(console\)](#)

Adicionar tags a um novo grupo (console)

Ao criar um grupo do X-Ray, você pode adicionar tags opcionais na página Criar grupo.

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. No painel de navegação, expanda Configuração e escolha Grupos.
3. Escolha Criar grupo.
4. Na página Criar grupo, especifique um nome e uma expressão de filtro para o grupo. Para obter mais informações sobre essas propriedades, consulte [Configurar grupos](#).
5. Em Tags, insira uma chave de tag e, opcionalmente, um valor de tag. Por exemplo, você pode inserir uma chave de tag **Stage** e um valor de tag **Production** para indicar que esse grupo é para uso em produção. Ao adicionar uma tag, uma nova linha aparece para você adicionar outra tag, se necessário. Consulte [Restrições de tag](#) neste tópico para saber sobre as limitações das tags.
6. Ao terminar de adicionar tags, escolha Criar grupo.

Adicionar tags a uma nova regra de amostragem (console)

Ao criar uma regra de amostragem do X-Ray, você pode adicionar tags na página Criar regra de amostragem.

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. No painel de navegação, expanda Configuração e escolha Amostragem.
3. Escolha Criar regra de amostragem.
4. Na página Criar regra de amostragem, especifique um nome, prioridade, limites, critérios de correspondência e atributos correspondentes. Para obter mais informações sobre essas propriedades, consulte [Configurar regras de amostragem](#).

5. Em Tags, insira uma chave de tag e, opcionalmente, um valor de tag. Por exemplo, você pode inserir uma chave de tag **Stage** e um valor de tag **Production** para indicar que essa regra de amostragem é para uso em produção. Ao adicionar uma tag, uma nova linha aparece para você adicionar outra tag, se necessário. Consulte [Restrições de tag](#) neste tópico para saber sobre as limitações das tags.
6. Escolha Criar regra de amostragem quando terminar de adicionar as tags.

Editar ou excluir tags de um grupo (console)

Você pode alterar ou excluir tags em um grupo do X-Ray na página Editar grupo.

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. No painel de navegação, expanda Configuração e escolha Grupos.
3. Na tabela Grupos, escolha o nome de um grupo.
4. Na página Editar grupo, em Tags, edite as chaves e os valores das tags. Não é possível ter chaves de tag duplicadas. Os valores das tags são opcionais; você pode excluir valores se quiser. Para obter mais informações sobre outras propriedades na página Editar grupo, consulte [Configurar grupos](#). Consulte [Restrições de tag](#) neste tópico para saber sobre as limitações das tags.
5. Para excluir uma tag, escolha X à direita da tag.
6. Ao terminar de editar ou excluir as tags, escolha Atualizar grupo.

Editar ou excluir tags de uma regra de amostragem (console)

Você pode alterar ou excluir tags em uma regra de amostragem do X-Ray na página Editar regra de amostragem.

1. Faça login no Console de gerenciamento da AWS e abra o console do X-Ray em <https://console.aws.amazon.com/xray/home>.
2. No painel de navegação, expanda Configuração e escolha Amostragem.
3. Na tabela Regras de amostragem, escolha o nome de uma regra de amostragem.
4. Em Tags, edite chaves e valores da tag. Não é possível ter chaves de tag duplicadas. Os valores das tags são opcionais; você pode excluir valores se quiser. Para obter mais informações sobre outras propriedades na página Editar regra de amostragem, consulte

[Configurar regras de amostragem](#). Consulte [Restrições de tag](#) neste tópico para saber sobre as limitações das tags.

5. Para excluir uma tag, escolha X à direita da tag.
6. Ao terminar de editar ou excluir as tags, escolha Atualizar regra de amostragem.

Gerenciar tags na AWS CLI

Você pode adicionar tags opcionais ao criar uma regra de amostragem ou um grupo do X-Ray. É possível usar a AWS CLI para criar e gerenciar tags. Para atualizar tags em um grupo ou regra de amostragem existente, use o console do AWS X-Ray ou as APIs [TagResource](#) ou [UntagResource](#).

Tópicos

- [Adicione tags a um novo grupo ou regra de amostragem do X-Ray \(CLI\)](#)
- [Adicionar tags a um recurso existente \(CLI\)](#)
- [Listar as tags em um recurso \(CLI\)](#)
- [Excluir tags de um recurso \(CLI\)](#)

Adicione tags a um novo grupo ou regra de amostragem do X-Ray (CLI)

Para adicionar tags opcionais ao criar um grupo ou regra de amostragem do X-Ray, use um dos comandos a seguir.

- Para adicionar tags a um novo grupo, execute o comando a seguir, substituindo *group_name* pelo nome do seu grupo, *mydomain.com* pelo endpoint do seu serviço, *key_name* por uma chave de tag e, opcionalmente, *value* por um valor de tag. Para obter mais informações sobre como criar um grupo, consulte [Grupos](#).

```
aws xray create-group \  
  --group-name "group_name" \  
  --filter-expression "service(\mydomain.com\") {fault OR error}" \  
  --tags [{"Key": "key_name", "Value": "value"}, {"Key": "key_name", "Value": "value"}]
```

Veja um exemplo a seguir.

```
aws xray create-group \  
  --group-name "AdminGroup" \  
  --filter-expression "service(\mydomain.com\") {fault OR error}" \  
  --tags [{"Key": "key_name", "Value": "value"}, {"Key": "key_name", "Value": "value"}]
```

```
--filter-expression "service(\"mydomain.com\") {fault OR error}" \  
--tags [{"Key": "Stage","Value": "Prod"}, {"Key": "Department","Value": "QA"}]
```

- Para adicionar tags a uma nova regra de amostragem, execute o comando a seguir, substituindo *key_name* por uma chave de tag e, opcionalmente, *value* por um valor de tag. Esse comando especifica os valores no parâmetro `--sampling-rule` como um arquivo JSON. Para obter mais informações sobre como criar uma regra de amostragem, consulte [Regras de amostragem](#).

```
aws xray create-sampling-rule \  
--cli-input-json file://file_name.json
```

Veja a seguir o conteúdo do arquivo JSON *file_name.json* especificado pelo parâmetro `--cli-input-json`.

```
{  
  "SamplingRule": {  
    "RuleName": "rule_name",  
    "RuleARN": "string",  
    "ResourceARN": "string",  
    "Priority": integer,  
    "FixedRate": double,  
    "ReservoirSize": integer,  
    "ServiceName": "string",  
    "ServiceType": "string",  
    "Host": "string",  
    "HTTPMethod": "string",  
    "URLPath": "string",  
    "Version": integer,  
    "Attributes": {"attribute_name": "value", "attribute_name": "value"...}  
  }  
  "Tags": [  
    {  
      "Key": "key_name",  
      "Value": "value"  
    },  
    {  
      "Key": "key_name",  
      "Value": "value"  
    }  
  ]  
}
```

O comando a seguir é um exemplo.

```
aws xray create-sampling-rule \  
  --cli-input-json file://9000-base-scorekeep.json
```

Veja a seguir o conteúdo do arquivo `9000-base-scorekeep.json` de exemplo especificado pelo parâmetro `--cli-input-json`.

```
{  
  "SamplingRule": {  
    "RuleName": "base-scorekeep",  
    "ResourceARN": "*",  
    "Priority": 9000,  
    "FixedRate": 0.1,  
    "ReservoirSize": 5,  
    "ServiceName": "Scorekeep",  
    "ServiceType": "*",  
    "Host": "*",  
    "HTTPMethod": "*",  
    "URLPath": "*",  
    "Version": 1  
  }  
  "Tags": [  
    {  
      "Key": "Stage",  
      "Value": "Prod"  
    },  
    {  
      "Key": "Department",  
      "Value": "QA"  
    }  
  ]  
}
```

Adicionar tags a um recurso existente (CLI)

Você pode executar o comando `tag-resource` para adicionar tags a um grupo ou regra de amostragem existente do X-Ray. Esse método pode ser mais simples do que adicionar tags executando `update-group` ou `update-sampling-rule`.

Para adicionar tags a um grupo ou a uma regra de amostragem, execute o comando a seguir, substituindo o ARN pelo ARN do recurso e especificando as chaves e os valores opcionais das tags que você deseja adicionar.

```
aws xray tag-resource \  
  --resource-arn "ARN" \  
  --tag-keys [{"Key": "key_name", "Value": "value"}, {"Key": "key_name", "Value": "value"}]
```

Veja um exemplo a seguir.

```
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"}]
```

Listar as tags em um recurso (CLI)

Você pode executar o comando `list-tags-for-resource` para listar as tags de um grupo ou regra de amostragem do X-Ray.

Para listar as tags associadas a um grupo ou a uma regra de amostragem, execute o comando a seguir, substituindo o ARN pelo ARN do recurso.

```
aws xray list-tags-for-resource \  
  --resource-arn "ARN"
```

Veja um exemplo a seguir.

```
aws xray list-tags-for-resource \  
  --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup"
```

Excluir tags de um recurso (CLI)

Você pode executar o comando `untag-resource` para remover as tags de um grupo ou regra de amostragem do X-Ray.

Para adicionar tags a um grupo ou a uma regra de amostragem, execute o comando a seguir, substituindo o ARN pelo ARN do recurso e especificando as chaves e os valores opcionais das tags que você deseja adicionar.

Você pode remover somente tags inteiras com o comando `untag-resource`. Para remover valores de tag, use o console do X-Ray ou exclua tags e adicione novas tags com as mesmas chaves, mas com valores diferentes ou vazios.

```
aws xray untag-resource \  
  --resource-arn "ARN" \  
  --tag-keys ["key_name", "key_name"]
```

Veja um exemplo a seguir.

```
aws xray untag-resource \  
  --resource-arn "arn:aws:xray:us-east-2:01234567890:group/group_name" \  
  --tag-keys ["Stage", "Department"]
```

Controlar o acesso a recursos do X-Ray com base em tags

Você pode anexar tags a grupos ou regras de amostragem do X-Ray ou passar tags em uma solicitação para o X-Ray. Para controlar o acesso baseado em tags, forneça informações sobre as tags no [elemento de condição](#) de uma política usando as chaves de condição `xray:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`. Para saber mais sobre essas chaves de condição, consulte [Controlar o acesso a recursos da AWS usando tags](#).

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte [Gerenciar o acesso a grupos e regras de amostragem do X-Ray com base em tags](#).

Solução de problemas do AWS X-Ray

Este tópico lista os erros e os problemas comuns que podem ser encontrados ao usar a API, o console ou os SDKs do X-Ray. Se encontrar um problema que não esteja listado aqui, você poderá usar o botão Feedback desta página para relatá-lo.

Seções

- [Mapa de rastreamento do X-Ray e páginas de detalhes do rastreamento](#)
- [X-Ray SDK para Java](#)
- [X-Ray SDK para Node.js](#)
- [O daemon do X-Ray](#)

Mapa de rastreamento do X-Ray e páginas de detalhes do rastreamento

As seções a seguir podem ajudar se você estiver tendo problemas ao usar o mapa de rastreamento do X-Ray e a página de detalhes do rastreamento:

Não vejo todos os meus logs do CloudWatch

A forma de configurar os logs para que eles apareçam no mapa de rastreamento do X-Ray e nas páginas de detalhes do rastreamento depende do serviço.

- Os logs do API Gateway serão exibidos se o registro estiver ativado no API Gateway.

Nem todos os nós do mapa de serviço oferecem suporte à visualização dos logs associados. Visualize os logs dos seguintes tipos de nós:

- Contexto do Lambda
- Função do Lambda
- Estágio da API Gateway
- Cluster do Amazon ECS
- Instância do Amazon ECS
- Serviço do Amazon ECS

- Tarefa do Amazon ECS
- Cluster do Amazon EKS
- Namespace do Amazon EKS
- Nó do Amazon EKS
- Pod do Amazon EKS
- Serviço do Amazon EKS

Não vejo todos os meus alarmes no mapa de rastreamento do X-Ray

O mapa de rastreamento do X-Ray mostrará somente o ícone de alerta de um nó se algum alarme associado a ele estiver no estado ALARME.

O mapa de rastreamento associa os alarmes aos nós usando a seguinte lógica:

- Se o nó representa um serviço da AWS, todos os alarmes com o namespace associado a esse serviço serão associados ao nó. Por exemplo, um nó do tipo `AWS::Kinesis` está vinculado a todos os alarmes baseados em métricas no namespace `AWS/Kinesis` do CloudWatch.
- Se o nó representar um recurso da AWS, os alarmes desse recurso específico serão vinculados. Por exemplo, um nó do tipo `AWS::DynamoDB::Table` com o nome "MyTable" está vinculado a todos os alarmes com base em uma métrica com o namespace `AWS/DynamoDB` e tem a dimensão `TableName` definida como `MyTable`.
- Se o nó for de tipo desconhecido, o que é identificado por uma borda tracejada ao redor do nome, nenhum alarme estará associado ao nó.

Não vejo alguns recursos da AWS no mapa de rastreamento

Nem todo recurso da AWS é representado por um nó dedicado. Alguns serviços da AWS são representados por um nó único para todas as solicitações ao serviço. Os seguintes tipos de recursos são exibidos com um nó por recurso:

- `AWS::DynamoDB::Table`
- `AWS::Lambda::Function`

As funções do Lambda são representadas por dois nós: um para o contêiner Lambda e outro para a função. Isso ajuda a identificar problemas de inicialização a frio com funções do Lambda. Os nós

de contêiner do Lambda são associados a alarmes e painéis da mesma maneira que os nós de função do Lambda.

- `AWS::ApiGateway::Stage`
- `AWS::SQS::Queue`
- `AWS::SNS::Topic`

Há muitos nós no mapa de rastreamento

Use grupos do X-Ray para dividir o mapa em diversos mapas. Para obter mais informações, consulte [Usar expressões de filtro com grupos](#).

X-Ray SDK para Java

Erro: exceção no thread "Thread-1" com `com.amazonaws.xray.exceptions.SegmentNotFoundException`: falha ao iniciar subsegmento chamado 'AmazonSNS': o segmento não pode ser encontrado.

Esse erro indica que o X-Ray SDK tentou registrar uma chamada de saída para a AWS, mas não foi possível encontrar um segmento aberto. Isso pode ocorrer nas seguintes situações:

- Um filtro de servlet não está configurado: o X-Ray SDK cria segmentos para as solicitações de entrada com um filtro chamado `AWSXRayServletFilter`. [Configure um filtro de servlet](#) para instrumentar as solicitações de entrada.
- Você está usando clientes instrumentados fora do código do servlet: se você usar um cliente instrumentado para fazer chamadas no código de inicialização ou em outro código que não seja executado em resposta a uma solicitação de entrada, deverá criar um segmento manualmente. Consulte [Instrumentar código de inicialização](#) para ver exemplos.
- Você está usando clientes instrumentados em threads de operador: quando você cria um thread, o gravador do X-Ray perde a referência do segmento aberto. Você pode usar os métodos [getTraceEntity](#) e [setTraceEntity](#) para obter uma referência ao segmento ou subsegmento atual ([Entity](#)) e transmiti-los de volta para o gravador dentro do thread. Para ver um exemplo, consulte [Usar clientes instrumentais em threads de operador](#).

X-Ray SDK para Node.js

Problema: o CLS não funciona com o Sequelize

Passe o namespace do X-Ray SDK para Node.js para o Sequelize com o método `cls`.

```
var AWSXRay = require('aws-xray-sdk');
const Sequelize = require('sequelize');
Sequelize.cls = AWSXRay.getNamespace();
const sequelize = new Sequelize(...);
```

Problema: o CLS não funciona com o Bluebird

Use `cls-bluebird` para fazer com que o Bluebird funcione com o CLS.

```
var AWSXRay = require('aws-xray-sdk');
var Promise = require('bluebird');
var clsBluebird = require('cls-bluebird');
clsBluebird(AWSXRay.getNamespace());
```

O daemon do X-Ray

Problema: O daemon está usando as credenciais erradas

O daemon usa o SDK da AWS para carregar as credenciais. Se você usar vários métodos para fornecer credenciais, o método com a precedência mais alta será usado. Consulte [Execução do daemon](#) para obter mais informações.

Histórico do documento para AWS X-Ray

A tabela a seguir descreve as mudanças importantes na documentação do AWS X-Ray. Para receber notificações sobre atualizações dessa documentação, é possível inscrever-se em um feed RSS.

Última atualização da documentação: 7 de março de 2024

Alteração	Descrição	Data
Cronograma de AWS suporte atualizado para X-Ray SDKs e Daemon	Em 25 de fevereiro de 2026, o AWS X-Ray SDKs/Daemon entrará no modo de manutenção, onde AWS limitará os lançamentos do X-Ray SDK e do Daemon para tratar apenas de problemas de segurança. Para obter mais informações sobre o cronograma de suporte, consulte o cronograma do X-Ray SDK e do Daemon Support. Recomendamos migrar para o. OpenTelemetry Para obter mais informações sobre a migração para OpenTelemetry, consulte Migrando da instrumentação X-Ray para a instrumentação OpenTelemetry	26 de novembro de 2025
end-of-supportAviso adicionado para AWS X-Ray SDKs e Daemon	Em 25 de fevereiro de 2027, o X-Ray descontinuará o suporte para AWS SDKs X-Ray e Daemon. Recomendamos migrar para o. OpenTelemetry Para obter	22 de agosto de 2025

mais informações, consulte [Migração da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

Adição de funcionalidade

Migrando do X-Ray para o. OpenTelemetry Para obter mais informações, consulte [Migração da instrumentação X-Ray para a instrumentação OpenTelemetry](#).

13 de junho de 2025

Adição de funcionalidade

AWS O X-Ray agora suporta a Pesquisa de Transações. Para obter mais informações, consulte [Transaction Search](#).

21 de novembro de 2024

Adição de funcionalidade

AWS O X-Ray agora oferece suporte ao Endpoint de OpenTelemetry Protocolo (OTLP). Para obter mais informações, consulte [OpenTelemetry](#).

21 de novembro de 2024

Adição de funcionalidade	O X-Ray agora registra eventos de dados PutTraceSegments GetTraceSummaries , incluindo , e BatchGetTraces para AWS CloudTrail. O X-Ray agora também registra o evento GetSamplingStatisticSummaries de gerenciamento em CloudTrail. Para obter mais informações, consulte Logging X-Ray API call with AWS CloudTrail.	7 de março de 2024
Adição de funcionalidade	O X-Ray agora suporta rastreamento IDs criado por meio de OpenTelemetry ou qualquer outra estrutura que esteja em conformidade com a especificação W3C Trace Context. Para obter mais informações, consulte Enviar dados de rastreamento para o X-Ray.	25 de outubro de 2023
Adição de funcionalidade	Agora você pode configurar o rastreamento ativo do Amazon SNS, que permite rastrear e analisar solicitações à medida que elas passam pelos tópicos do Amazon SNS. Para obter mais informações, consulte Amazon SNS e AWS X-Ray	8 de fevereiro de 2023

[Tópico atualizado do X-Ray SDK para Node.js](#)

Detalhes adicionados para instrumentar clientes usando o AWS SDK para JavaScript V3. Para obter detalhes, consulte [Rastreamento de chamadas AWS do SDK com o X-Ray SDK](#) para Node.js.

7 de fevereiro de 2023

[Detalhes atualizados da política gerenciada do IAM](#)

Foi adicionada permissão do IAM para observabilidade entre contas às políticas gerenciadas `AWSXRayReadOnlyAccess` , `AWSXRayFullAccess` e `AWSXrayCrossAccountSharingConfiguration` . Para obter detalhes, consulte [Políticas gerenciadas do IAM para X-Ray](#).

7 de fevereiro de 2023

[Adição de funcionalidade](#)

AWS X-Ray agora oferece suporte à observabilidade entre contas, permitindo o monitorar e solucionar problemas de aplicativos que abrangem várias contas em um. Região da AWS Para obter detalhes, consulte [Rastreamento entre contas](#).

27 de novembro de 2022

[Adição de funcionalidade](#)

Agora você pode visualizar rastreamentos vinculados entre produtores de mensagens, uma fila do Amazon SQS e consumidores e ter uma visão conectada dos rastreamentos enviados de aplicativos orientados a eventos. Para obter mais informações, consulte [Rastrear aplicativos orientados a eventos](#).

20 de novembro de 2022

[Detalhes atualizados da política gerenciada do IAM](#)

Foi adicionada a permissão do IAM para listar políticas de recursos na política gerenciada `AWSXRayReadOnlyAccess`. Para obter detalhes, consulte [Políticas gerenciadas do IAM para X-Ray](#).

15 de novembro de 2022

[Atualização de permissões do console do IAM e de detalhes de política gerenciada](#)

O conjunto de permissões do IAM que o console do X-Ray usa foi atualizado, bem como a descrição da política gerenciada `AWSXRayReadOnlyAccess`. Para obter detalhes, consulte [Usar o console do X-Ray](#).

11 de novembro de 2022

[AWS Distro adicionada para Ruby OpenTelemetry](#)

AWS O Distro for OpenTelemetry (ADOT) fornece um único conjunto de código aberto APIs, bibliotecas e agentes para coletar rastreamentos e métricas distribuídos. O ADOT Ruby permite que você instrumente um aplicativo Ruby para o X-Ray e outros back-ends de rastreamento. Para obter mais informações, consulte [AWS Distro for OpenTelemetry Ruby](#).

7 de fevereiro de 2022

[Adição de funcionalidade](#)

Agora você pode visualizar rastreamentos e configurar o X-Ray a partir do CloudWatch console. Para obter mais informações, consulte [Console do X-Ray](#).

24 de janeiro de 2022

[CloudWatch RUM integrado](#)

Com AWS X-Ray o CloudWatch RUM, você pode analisar e depurar o caminho da solicitação, começando pelos usuários finais do seu aplicativo por meio de serviços AWS gerenciados downstream. Para obter mais informações, consulte [CloudWatch RUM AWS X-Ray e](#).

3 de dezembro de 2021

[AWS Distribuição integrada para OpenTelemetry](#)

O AWS Distro for OpenTelemetry (ADOT) fornece um único conjunto de código aberto APIs, bibliotecas e agentes para coletar rastreamentos e métricas distribuídos. O ADOT permite que você instrumente um aplicativo para X-Ray e outros back-ends de rastreamento. Para obter mais informações, consulte [Instrumentar seu aplicativo](#).

23 de setembro de 2021

[Adição de funcionalidade](#)

AWS X-Ray agora se integra à Amazon Virtual Private Cloud, permitindo que recursos em sua Amazon VPC se comuniquem com o serviço X-Ray sem passar pela Internet pública. Para obter mais informações, consulte [Como usar AWS X-Ray com VPC endpoints](#).

20 de maio de 2021

[Adição de funcionalidade](#)

AWS X-Ray agora se integra CloudFormation, permitindo o que você provisione e configure recursos de X-Ray. Para obter mais informações, consulte [Creating X-Ray resources with CloudFormation](#).

6 de maio de 2021

Adição de funcionalidade	AWS X-Ray agora se integra à Amazon EventBridge para rastrear eventos que são transmitidos EventBridge. Isso fornece aos usuários uma visão mais completa do sistema. Para obter mais informações, consulte Amazon EventBridge AWS X-Ray e .	2 de março de 2021
Daemon adicionado ao ECR	Agora o daemon pode ser baixado do Amazon ECR. Para obter mais informações, consulte o Baixar o daemon .	1.º de março de 2021
Adição de funcionalidade	AWS X-Ray agora oferece suporte a notificações relacionadas a insights para a Amazon EventBridge. Isso permite que você execute ações automáticas sobre o uso de insights EventBridge. Para obter mais informações, consulte Notificações de insights .	15 de outubro de 2020
Daemons adicionados para baixar	AWS X-Ray introduz o daemon de suporte para Linux. ARM64 Para obter mais informações, consulte AWS X-Ray daemonbrazil ws	1.º de outubro de 2020

Adição de funcionalidade

AWS X-Ray agora oferece suporte à integração ativa com o Amazon CloudWatch Synthetics. Isso permite que você veja detalhes sobre um nó de cliente canário do Synthetics, como tempo de resposta e status. Você também pode realizar análises no console do Analytics com base nas informações de um nó de cliente canário do Synthetics. Para obter mais informações, consulte [Depuração de CloudWatch canários sintéticos usando X-Ray](#).

24 de setembro de 2020

Adição de funcionalidade

AWS X-Ray agora oferece suporte ao rastreamento de end-to-end fluxos de trabalho para AWS Step Functions. Você pode visualizar os componentes do aplicativo, identificar gargalos de performance e solucionar problemas de solicitações que resultaram em um erro. Para obter mais informações, consulte [AWS Step Functions](#) e [AWS X-Ray](#).

14 de setembro de 2020

Adição de funcionalidade

AWS X-Ray apresenta insights para analisar continuamente os dados de rastreamento em sua conta para identificar problemas emergentes em seus aplicativos. Os insights registram incidentes e rastreiam o impacto do incidente até a resolução. Para obter mais informações, consulte [Usando insights no AWS X-Ray console](#)

3 de setembro de 2020

Adição de funcionalidade

AWS X-Ray apresenta o agente de instrumentação automática Java, permitindo que os clientes colem dados de rastreamento sem precisar modificar o aplicativo existente baseado em Java. Agora você pode rastrear aplicativos Java baseados na web e em servlet com o mínimo de alteração na configuração e sem alteração de código. Para obter mais informações, consulte [Agente de instrumentação automática do AWS X-Ray para Java](#).

3 de setembro de 2020

Adição de funcionalidade

AWS X-Ray adicionou uma nova página de grupos ao console X-Ray para ajudar a facilitar a criação e o gerenciamento de grupos de rastreamentos. Para obter mais informações, consulte [Configurar grupos no console do X-Ray](#).

24 de agosto de 2020

Adição de funcionalidade

AWS X-Ray agora permite adicionar tags a grupos e regras de amostragem. Também é possível controlar o acesso a grupos e regras de amostragem com base em tags. Para obter mais informações, consulte [Marcar regras de amostragem e grupos do X-Ray](#) e [Gerenciar o acesso a grupos e regras de amostragem do X-Ray com base em tags](#).

24 de agosto de 2020

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.