aws

Guida per gli sviluppatori

AWS X-Ray



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

AWS X-Ray: Guida per gli sviluppatori

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

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Cosa è AWS X-Ray? 1
Nozioni di base 4
Scelta di un'interfaccia 6
Usa un SDK 8
Usa il ADOT SDK 9
Usa il X-Ray SDK 10
Usa una console 12
Usa la CloudWatch console Amazon 12
Usa la console X-Ray 13
Esplora la console X-Ray 14
Mappa di tracciamento 15
Tracce 21
Espressioni filtro 31
Cross-account tracciamento 44
Tracciamento delle applicazioni basate sugli eventi 47
Istogrammi 51
Approfondimenti 53
Analisi 60
Gruppi 68
Campionamento 78
Campionamento adattivo 85
Deep linking tra console 95
Usa l' X-Ray API 97
Tutorial 99
Invio dei dati 104
Ottenimento dei dati 109
Configurazione 123
Campionamento 131
Documenti di un segmento 135
Concetti 155
Segmenti 155
Sottosegmenti 156
Grafico del servizio 159
Tracce 161

Campionamento ... 162
Intestazione di tracciamento ... 163
Espressioni filtro ... 164
Gruppi ... 165
Annotazioni e metadati ... 165
Errori, malfunzionamenti ed eccezioni ... 166
Sicurezza ... 167
Protezione dei dati ... 167
Gestione dell'identità e degli accessi ... 170
Destinatari ... 170
Autenticazione con identità ... 171
Gestione dell'accesso tramite policy ... 172
Come AWS X-Ray funziona con IAM ... 174
Esempi di policy basate su identità ... 181
Risoluzione dei problemi ... 192
Registrazione di log e monitoraggio ... 195
Convalida della conformità ... 196
Resilienza ... 196
Sicurezza dell'infrastruttura ... 196
Endpoint VPC ... 197
Creazione di un endpoint VPC per X-Ray ... 197
Controllo dell'accesso al tuo X-Ray endpoint VPC ... 199
Regioni supportate ... 200
Prevenzione del confused deputy tra servizi ... 201
Applicazione di esempio ... 204
Tutorial Scorekeep ... 206
Prerequisiti ... 207
Installa l'applicazione Scorekeep utilizzando CloudFormation ... 208
Generare i dati di tracciamento ... 209
Visualizza la mappa delle tracce nel Console di gestione AWS ... 210
Configurazione delle notifiche Amazon SNS ... 218
Esplorare l'applicazione di esempio ... 219
Opzionale: policy con privilegio minimo ... 224
Eliminazione ... 227
Fasi successive ... 228
AWS Client SDK ... 228

Sottosegmenti personalizzati 229
Annotazioni e metadati 230
Client HTTP 231
Client SQL 232
AWS Lambda funzioni 235
Nome casuale 236
Worker 238
Analisi del codice di avvio 241
Analisi degli script 243
Strumentazione dei client web 245
Thread worker 249
Demone X-Ray 252
Download del daemon 253
Verifica della firma dell'archivio del daemon 254
Esecuzione del daemon 255
Dare al demone il permesso di inviare dati a X-Ray 256
Registri dei daemon X-Ray 256
Configurazione 257
Variabili di ambiente supportate 258
Utilizzo delle opzioni della riga di comando 258
Utilizzo di un file di configurazione 260
Esecuzione del daemon in locale 261
Esecuzione del demone X-Ray su Linux 262
Esecuzione del demone X-Ray in un contenitore Docker 262
Esecuzione del demone X-Ray su Windows 264
Esecuzione del demone X-Ray su OS X 265
Su Elastic Beanstalk 265
Utilizzo dell'integrazione X-Ray di Elastic Beanstalk per eseguire il demone X-Ray 266
Scaricamento ed esecuzione manuale del demone X-Ray (avanzato) 268
Su Amazon EC2 270
Su Amazon ECS 271
Utilizzo dell'immagine Docker ufficiale 272
Creare e compilare un'immagine Docker 272
Configura le opzioni della riga di comando nella console Amazon ECS 275
Integrazione con Servizi AWS 277
Amazon Bedrock AgentCore 279

Simple Storage Service (Amazon S3) 280
Simple Storage Service (Amazon S3) 280
Amazon EC2 280
Amazon SNS 280
Configura il tracciamento attivo di Amazon SNS 281
Visualizza le tracce degli editori e degli abbonati di Amazon SNS nella console X-Ray 283
Amazon SQS 284
Invio dell'intestazione di traccia HTTP 286
Recupero dell'intestazione di traccia e recupero del contesto di traccia 286
Amazon S3 287
Configurazione delle notifiche degli eventi di Amazon S3 288
AWS Distro per OpenTelemetry 289
AWS Distro per OpenTelemetry 289
AWS Config 290
Creazione di un trigger di funzione Lambda 290
Creazione di una personalizzazione AWS Config regola per la radiografia 292
Risultati di esempio 293
Notifiche Amazon SNS 293
AWS AppSync 293
API Gateway 294
App Mesh 296
App Runner 299
CloudTrail 299
Eventi di gestione dei raggi X in CloudTrail 301
Eventi relativi ai dati a raggi X in CloudTrail 302
Esempi di eventi X-Ray 303
CloudWatch 306
CloudWatch RUM 306
CloudWatch Synthetics 307
Elastic Beanstalk 315
Elastic Load Balancing 315
EventBridge 316
Visualizzazione dell'origine e delle destinazioni sulla mappa del X-Ray servizio 316
Propaga il contesto di traccia alle destinazioni degli eventi 317
Lambda 323
Funzioni Step 325

Strumentazione della vostra applicazione 327
Strumentate la vostra applicazione con Distro per AWS OpenTelemetry 328
Strumentate la vostra applicazione con AWS X-Ray SDKs 329
Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs 330
Transaction Search 332
OpenTelemetry Endpoint del protocollo (OTLP) 333
Utilizzo di Go 334
AWS Distro for Go OpenTelemetry 334
X-Ray SDK for Go 335
Requisiti 337
Documentazione di riferimento 337
Configurazione 337
Richieste in entrata 344
AWS Client SDK 347
Chiamate HTTP in uscita 349
Query SQL 350
Sottosegmenti personalizzati 351
Annotazioni e metadati 352
Utilizzo di Java 355
AWS Distro per Java OpenTelemetry 355
X-Ray SDK per Java 356
Sottomoduli 358
Requisiti 359
Gestione delle dipendenze 359
Agente di strumentazione automatica 361
Configurazione 373
Richieste in entrata 385
AWS Client SDK 390
Chiamate HTTP in uscita 393
Query SQL 395
Sottosegmenti personalizzati 398
Annotazioni e metadati 401
Monitoraggio 406
Multithreading 410
AOP con Spring 412
Utilizzo di Node.js 417

AWS Distro per OpenTelemetry JavaScript ... 417
X-Ray SDK per Node.js ... 418
Requisiti ... 420
Gestione delle dipendenze ... 421
Esempi di Node.js ... 421
Configurazione ... 422
Richieste in entrata ... 427
AWS Client SDK ... 431
Chiamate HTTP in uscita ... 435
Query SQL ... 438
Sottosegmenti personalizzati ... 439
Annotazioni e metadati ... 442
Uso di Python ... 447
AWS Distro per Python OpenTelemetry ... 447
X-Ray SDK per Python ... 448
Requisiti ... 450
Gestione delle dipendenze ... 451
Configurazione ... 451
Richieste in entrata ... 459
Applicare le patch alle librerie ... 465
AWS Client SDK ... 468
Chiamate HTTP in uscita ... 469
Sottosegmenti personalizzati ... 471
Annotazioni e metadati ... 473
Analizzare applicazioni serverless ... 477
Utilizzo di .NET ... 484
AWS Distro per.NET OpenTelemetry ... 484
X-Ray SDK per.NET ... 485
Requisiti ... 487
Aggiungere l'X-Ray SDK for .NET all'applicazione ... 487
Gestione delle dipendenze ... 487
Configurazione ... 489
Richieste in entrata ... 496
AWS Client SDK ... 501
Chiamate HTTP in uscita ... 503
Query SQL ... 505

Sottosegmenti personalizzati 509
Annotazioni e metadati 510
Lavorare con Ruby 514
AWS Distro per Ruby OpenTelemetry 514
X-Ray SDK per Ruby 515
Requisiti 516
Configurazione 517
Richieste in entrata 524
Applicare le patch alle librerie 528
AWS Client SDK 529
Sottosegmenti personalizzati 530
Annotazioni e metadati 531
Cronologia di X-Ray SDK e Daemon Support 536
Migrazione dalla strumentazione a raggi X alla strumentazione OpenTelemetry 537
Comprensione OpenTelemetry 538
OpenTelemetry supporto in AWS 538
Comprensione dei concetti relativi alla migrazione OpenTelemetry 539
Confronto delle funzionalità 540
Impostazione e configurazione del tracciamento 541
Rilevamento delle risorse nell'ambiente 542
Gestione delle strategie di campionamento 543
Gestione del contesto di traccia 544
Propagazione del contesto di traccia 544
Utilizzo della strumentazione della libreria 545
Esportazione di tracce 545
Elaborazione e inoltro delle tracce 546
Elaborazione Span (concetto specifico) OpenTelemetry 546
Bagaglio (concetto specifico) OpenTelemetry 546
Panoramica sulla migrazione 547
Raccomandazioni per applicazioni nuove ed esistenti 547
Tracciamento delle modifiche alla configurazione 548
Modifiche alla strumentazione della libreria 549
Modifiche alla strumentazione dell'ambiente Lambda 549
Creazione manuale di dati di traccia 549
Migrazione da X-Ray Daemon AWS CloudWatch a un agente o un raccoglitore OpenTelemetry 550

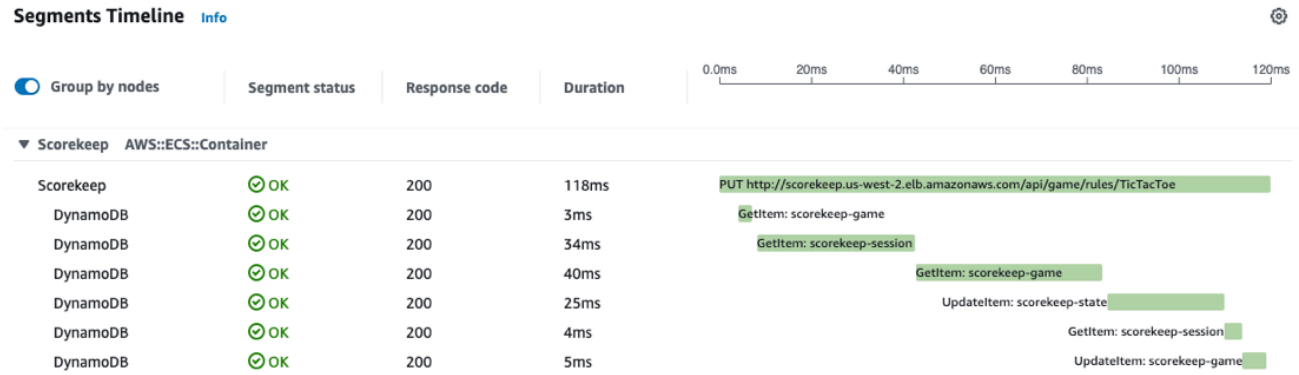
Migrazione su Amazon EC2 o server locali 551
Migrazione su Amazon ECS 554
Migrazione su Elastic Beanstalk 559
OpenTelemetry Migrazione a Java 560
Soluzione di strumentazione automatica a codice zero 560
Soluzioni di strumentazione manuale con SDK 561
Tracciamento delle richieste in arrivo (strumentazione Spring Framework) 564
AWS Strumentazione SDK v2 565
Analisi delle chiamate HTTP in uscita 566
Supporto strumentale per altre librerie 568
Creazione manuale dei dati di traccia 568
Strumentazione Lambda 571
Esegui la migrazione a OpenTelemetry Go 576
Strumentazione manuale con l'SDK 576
Tracciamento delle richieste in arrivo (strumentazione del gestore HTTP) 578
AWS SDK per strumentazione Go v2 579
Analisi delle chiamate HTTP in uscita 581
Supporto alla strumentazione per altre librerie 582
Creazione manuale di dati di traccia 582
Strumentazione manuale Lambda 584
Migrare a Node.js OpenTelemetry 590
Soluzioni di strumentazione automatica a codice zero 591
Soluzioni di strumentazione manuale 591
Tracciamento delle richieste in arrivo 594
AWS Strumentazione SDK V3 JavaScript 579
Analisi delle chiamate HTTP in uscita 598
Supporto alla strumentazione per altre librerie 599
Creazione manuale di dati di traccia 582
Strumentazione Lambda 584
Esegui la migrazione a.NET OpenTelemetry 602
Soluzioni di strumentazione automatica a codice zero 603
Soluzioni di strumentazione manuale con SDK 604
Creazione manuale di dati di traccia 607
Tracciamento delle richieste in arrivo (strumentazione di base ASP.NET e ASP.NET) 610
AWS Strumentazione SDK 611
Analisi delle chiamate HTTP in uscita 612

Supporto alla strumentazione per altre librerie ... 612
Strumentazione Lambda ... 584
Migrare a Python OpenTelemetry ... 617
Soluzioni di strumentazione automatica a codice zero ... 618
Strumenta manualmente le tue applicazioni ... 618
Tracciamento dell'inizializzazione della configurazione ... 619
Tracciamento delle richieste in arrivo ... 622
AWS Strumentazione SDK ... 623
Strumentazione delle chiamate HTTP in uscita tramite richieste ... 625
Supporto strumentale per altre librerie ... 626
Creazione manuale di dati di traccia ... 626
Strumentazione Lambda ... 628
Migra a Ruby OpenTelemetry ... 629
Strumenta manualmente le tue soluzioni con l'SDK ... 630
Tracciamento delle richieste in arrivo (strumentazione Rails) ... 632
AWS Strumentazione SDK ... 633
Analisi delle chiamate HTTP in uscita ... 634
Supporto alla strumentazione per altre librerie ... 635
Creazione manuale dei dati di traccia ... 635
Strumentazione manuale Lambda ... 638
Creazione di risorse X-Ray con CloudFormation ... 641
X-Ray e modelli CloudFormation ... 641
Scopri di più su CloudFormation ... 641
Assegnazione di tag ... 642
Limitazioni applicate ai tag ... 643
Gestione dei tag nella console ... 643
Aggiungi tag a un nuovo gruppo (console) ... 644
Aggiungi tag a una nuova regola di campionamento (console) ... 644
Modifica o elimina i tag per un gruppo (console) ... 645
Modifica o elimina i tag per una regola di campionamento (console) ... 645
Gestione dei tag in AWS CLI ... 646
Aggiungere tag a un nuovo gruppo X-Ray o a una nuova regola di campionamento (CLI) ... 646
Aggiungere tag a una risorsa esistente (CLI) ... 649
Elenca i tag su una risorsa (CLI) ... 649
Eliminare i tag su una risorsa (CLI) ... 649
Controlla l'accesso alle risorse X-Ray in base ai tag ... 650

Risoluzione dei problemi 651
Mappa di tracciamento a raggi X e pagine con i dettagli della traccia 651
Non vedo tutti i miei registri CloudWatch 651
Non vedo tutti i miei allarmi sulla mappa di tracciamento a raggi X 652
Non vedo alcune AWS risorse sulla mappa di tracciamento 652
Ci sono troppi nodi sulla mappa di traccia 653
X-Ray SDK per Java 653
X-Ray SDK per Node.js 653
Il demone X-Ray 654
Cronologia dei documenti 655
..... dclxv

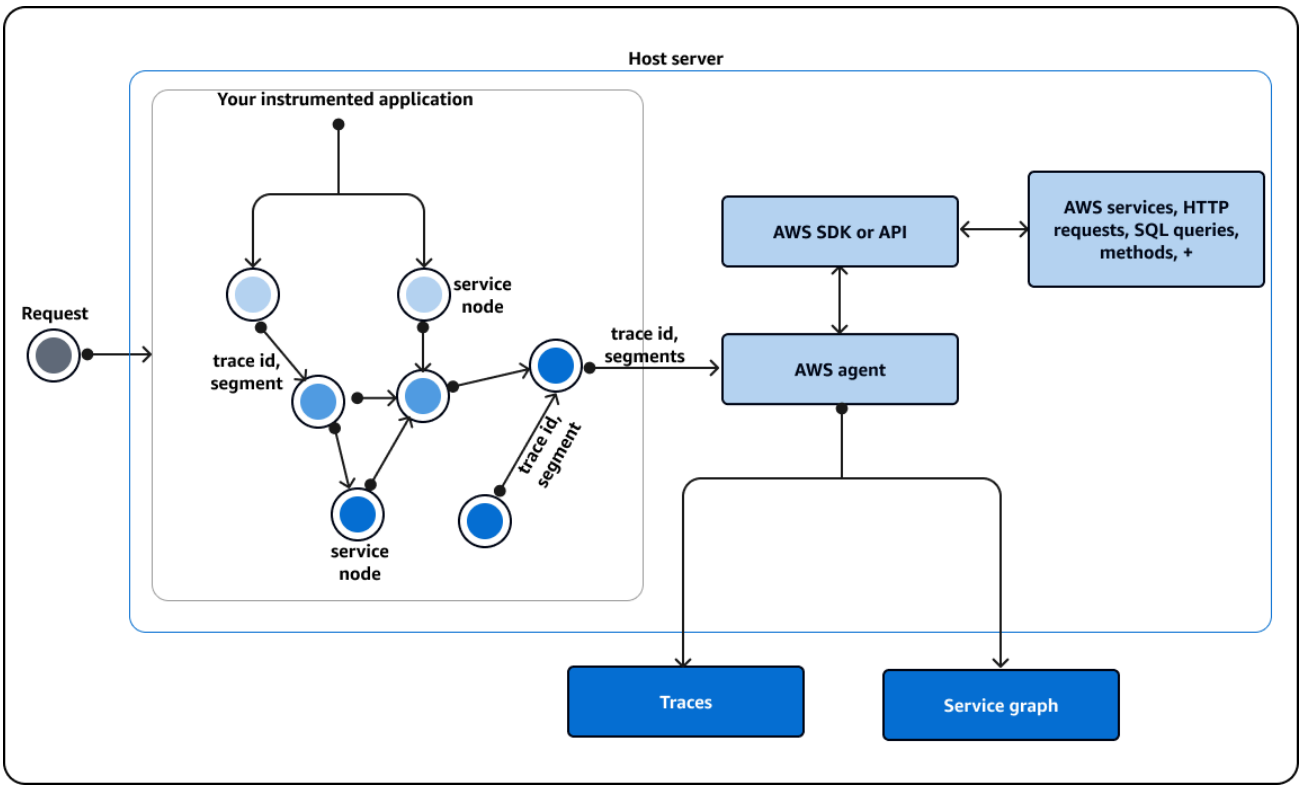
Cosa è AWS X-Ray?

AWS X-Ray è un servizio che raccoglie dati sulle richieste soddisfatte dall'applicazione e fornisce strumenti che è possibile utilizzare per visualizzare, filtrare e ottenere informazioni dettagliate su tali dati per identificare problemi e opportunità di ottimizzazione. Per qualsiasi richiesta tracciata all'applicazione, puoi visualizzare informazioni dettagliate non solo sulla richiesta e sulla risposta, ma anche sulle chiamate effettuate dall'applicazione alle AWS risorse di downstream, ai microservizi, ai database e alle API web.



AWS X-Ray riceve tracce dall'applicazione, oltre agli Servizi AWS usi dell'applicazione già integrati con X-Ray. La strumentazione dell'applicazione implica l'invio di dati di traccia per le richieste in entrata e in uscita e altri eventi all'interno dell'applicazione, insieme ai metadati relativi a ciascuna richiesta. Molti scenari di analisi richiedono solo modifiche alla configurazione. Ad esempio, è possibile strumentare tutte le richieste HTTP in entrata e le chiamate downstream effettuate dall'applicazione Java. Servizi AWS Esistono diversi SDK, agenti e strumenti che possono essere utilizzati per strumentare l'applicazione per il tracciamento. X-Ray Per ulteriori informazioni, consulta Strumentazione dell'applicazione.

Servizi AWS integrati con X-Ray possono aggiungere intestazioni di tracciamento alle richieste in arrivo, inviare dati di traccia o eseguire il demone X-Ray. X-Ray Ad esempio, AWS Lambda puoi inviare dati di traccia sulle richieste alle tue funzioni Lambda ed eseguire il X-Ray demone sui lavoratori per semplificare l'uso dell'SDK. X-Ray



Invece di inviare i dati di tracciamento direttamente a X-Ray, ogni SDK client invia i documenti del segmento JSON a un processo daemon in ascolto del traffico UDP. Il X-Ray demone memorizza nel buffer i segmenti in una coda e li carica in batch. X-Ray Il demone è disponibile per Linux, Windows e macOS ed è incluso nelle piattaforme. AWS Elastic Beanstalk AWS Lambda

X-Ray utilizza i dati di tracciamento provenienti dalle AWS risorse che alimentano le tue applicazioni cloud per generare una mappa di traccia dettagliata. La mappa di tracciamento mostra il client, il servizio front-end e i servizi di backend che il servizio front-end chiama per elaborare le richieste e conservare i dati. Usa la mappa di tracciamento per identificare colli di bottiglia, picchi di latenza e altri problemi da risolvere per migliorare le prestazioni delle tue applicazioni.


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

Guida introduttiva con X-Ray

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026 AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, che AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

Per utilizzarlo, procedi nel seguente X-Ray modo:

1. Strumenta la tua applicazione, che consente di X-Ray tenere traccia del modo in cui l'applicazione elabora una richiesta.

   - Usa gli X-Ray SDK, le X-Ray API ADOT o i segnali CloudWatch applicativi a cui inviare dati di traccia. X-Ray Per ulteriori informazioni sull'interfaccia da utilizzare, vedere. Scelta di un'interfaccia

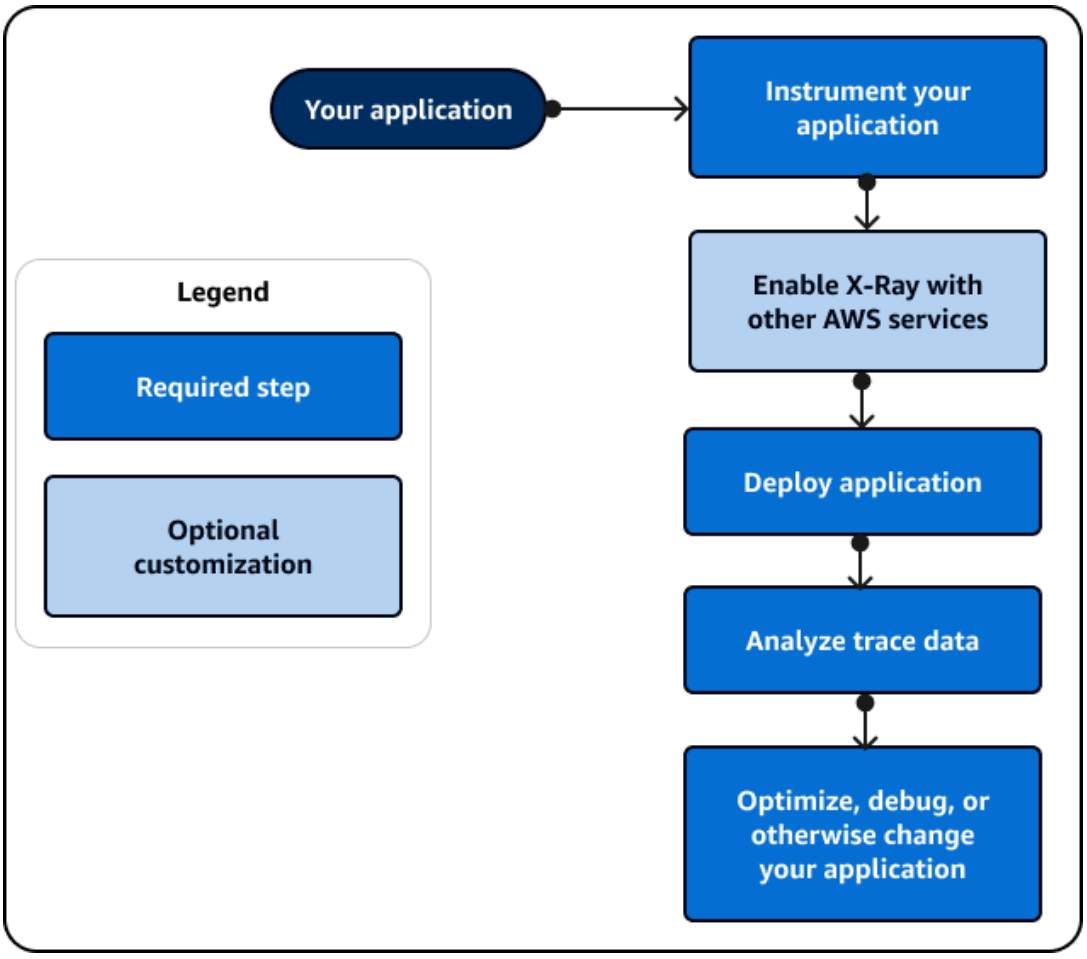
   Per ulteriori informazioni sulla strumentazione, vedere. Strumentazione della tua applicazione per AWS X-Ray
2. (Facoltativo) Configura X-Ray per lavorare con altri Servizi AWS che si integrano con. X-Ray È possibile campionare le tracce e aggiungere intestazioni alle richieste in arrivo, eseguire un agente o un raccoglitore e inviare automaticamente i dati di traccia a. X-Ray Per ulteriori informazioni, consulta Integrazione AWS X-Ray con altri Servizi AWS.
3. Implementa la tua applicazione strumentata. Man mano che l'applicazione riceve le richieste, l' X-Ray SDK registrerà i dati di traccia, segmento e sottosegmento. In questo passaggio, potresti anche dover impostare una policy IAM e distribuire un agente o un raccoglitore.

   - Ad esempio, gli script per distribuire un'applicazione utilizzando l'SDK AWS Distro for OpenTelemetry (ADOT) e l' CloudWatch agente su piattaforme diverse, vedi Application Signals Demo Scripts. https://github.com/aws-observability/application-signals-demo/tree/main/scripts

- Per uno script di esempio per distribuire un'applicazione utilizzando l'SDK e il X-Ray demone, vedere. X-Ray AWS X-Ray applicazione di esempio

4. (Facoltativo) Apri una console per visualizzare e analizzare i dati. È possibile visualizzare una rappresentazione GUI di una mappa di tracciamento, una mappa dei servizi e altro ancora per verificare il funzionamento dell'applicazione. Usa le informazioni grafiche nella console per ottimizzare, eseguire il debug e comprendere la tua applicazione. Per ulteriori informazioni sulla scelta di una console, vedere. Usa una console

Il diagramma seguente mostra come iniziare a utilizzare X-Ray:



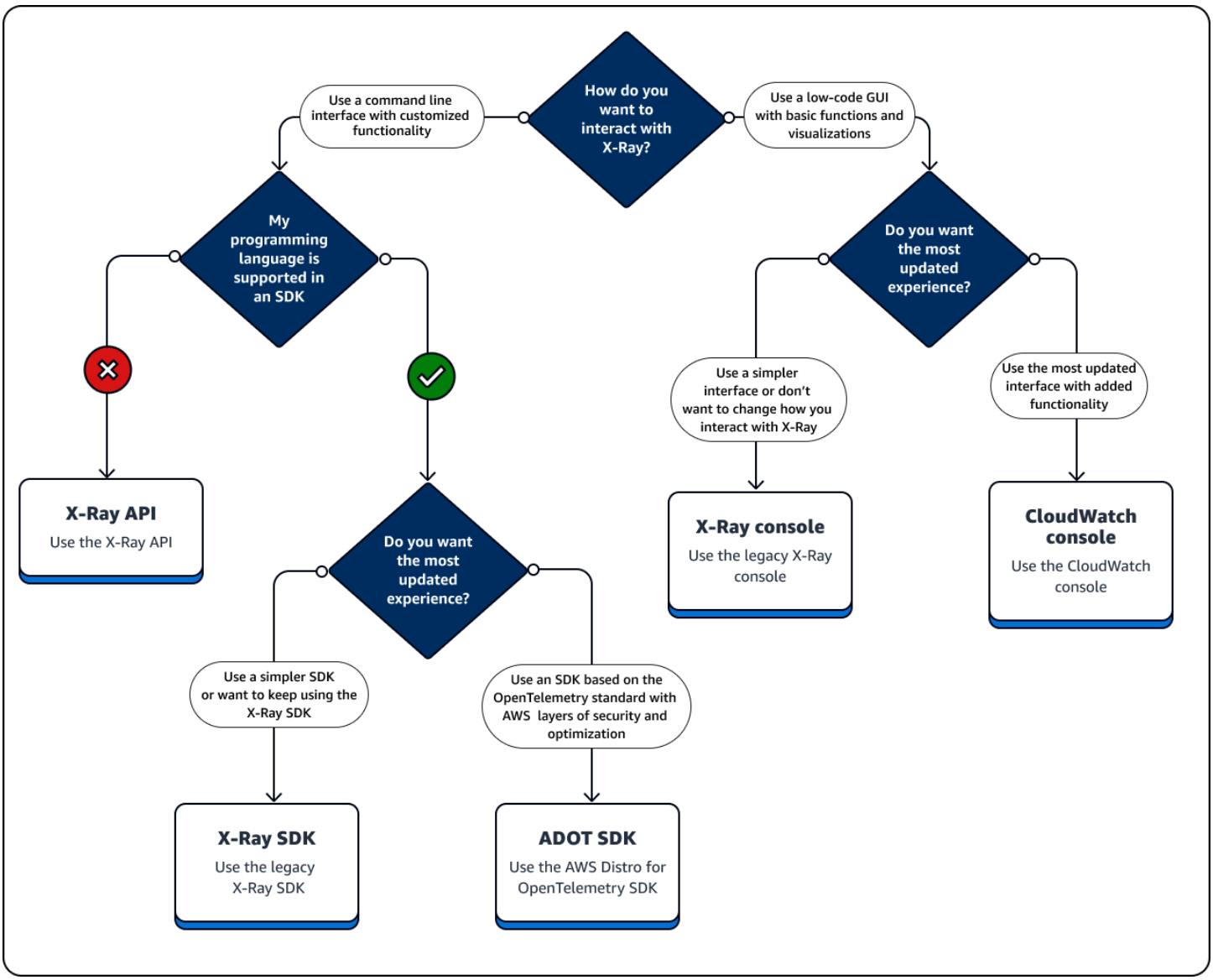
Per un esempio dei dati e delle mappe disponibili nella console, avvia un'applicazione di esempio già dotata degli strumenti per generare dati di traccia. In pochi minuti, è possibile generare traffico, inviare segmenti e visualizzare una X-Ray mappa di tracciamento e servizi.

Scelta di un'interfaccia

AWS X-Ray può fornire informazioni su come funziona l'applicazione e su come interagisce con altri servizi e risorse. Dopo aver strumentato o configurato l'applicazione, X-Ray raccoglie i dati di tracciamento man mano che l'applicazione soddisfa le richieste. È possibile analizzare questi dati di traccia per identificare i problemi di prestazioni, risolvere gli errori e ottimizzare le risorse. Questa guida mostra come interagire X-Ray con le seguenti linee guida:

- Usa un Console di gestione AWS se vuoi iniziare rapidamente o puoi utilizzare visualizzazioni predefinite per eseguire attività di base.
  - Scegli la CloudWatch console Amazon per l'esperienza utente più aggiornata che contiene tutte le funzionalità della X-Ray console.
  - Usa la X-Ray console se desideri un'interfaccia più semplice o non vuoi cambiare il modo in cui interagisci X-Ray.
- Usa un SDK se hai bisogno di più funzionalità personalizzate di tracciamento, monitoraggio o registrazione di quante ne possa fornire un an Console di gestione AWS .
  - Scegli l'ADOTSDK se desideri un SDK indipendente dal fornitore basato sull'SDK open source OpenTelemetry con livelli aggiuntivi di sicurezza e ottimizzazione. AWS
  - Scegli l' X-Ray SDK se desideri un SDK più semplice o non desideri aggiornare il codice dell'applicazione.
- Usa le operazioni X-Ray API se un SDK non supporta il linguaggio di programmazione dell'applicazione.

Il diagramma seguente ti aiuta a scegliere come interagire con X-Ray:



Esplora i tipi di interfaccia

- Usa un SDK
- Usa una console
- Usa l' X-Ray API

Usa un SDK

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

Usa un SDK se desideri utilizzare un'interfaccia a riga di comando o hai bisogno di più funzionalità di tracciamento, monitoraggio o registrazione personalizzate rispetto a quelle disponibili in un. Console di gestione AWS Puoi anche utilizzare un AWS SDK per sviluppare programmi che utilizzano le API. X-Ray Puoi utilizzare l'SDK AWS Distro for OpenTelemetry (ADOT) o l'SDK. X-Ray

Se si utilizza un SDK, è possibile aggiungere personalizzazioni al flusso di lavoro sia quando si utilizza lo strumento dell'applicazione sia quando si configura il raccoglitore o l'agente. Puoi utilizzare un SDK per eseguire le seguenti attività che non puoi eseguire utilizzando un: Console di gestione AWS

- Pubblica metriche personalizzate: campiona metriche ad alta risoluzione fino a 1 secondo, utilizza più dimensioni per aggiungere informazioni su una metrica e aggrega i punti dati in un set di statistiche.
- Personalizza il tuo raccoglitore: personalizza la configurazione per qualsiasi parte di un raccoglitore, inclusi ricevitore, processore, esportatore e connettore.
- Personalizza la tua strumentazione: personalizza segmenti e sottosegmenti, aggiungi coppie chiave-valore personalizzate come attributi e crea metriche personalizzate.
- Crea e aggiorna le regole di campionamento a livello di codice.

Usa l'ADOTSDK se desideri la flessibilità di utilizzare un OpenTelemetry SDK standardizzato con livelli aggiuntivi di sicurezza e ottimizzazione. AWS L'SDK AWS Distro for OpenTelemetry (ADOT) è un pacchetto indipendente dal fornitore che consente l'integrazione con backend di altri fornitori e non servizi senza dover riprogrammare il codice.AWS

Usa l' X-Ray SDK se stai già utilizzando l' X-Ray SDK, esegui l'integrazione solo con i AWS backend e non vuoi modificare il modo in cui interagisci o il codice dell'applicazione. X-Ray

Per ulteriori informazioni su ciascuna funzionalità, consulta. Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs

Usa il ADOT SDK

L'ADOTSDK è un set di API, librerie e agenti open source che inviano dati ai servizi di backend. ADOTè supportato da AWS, si integra con più backend e agenti e fornisce un gran numero di librerie open source gestite dalla comunità. OpenTelemetry Usa l'ADOTSDK per strumentare la tua applicazione e raccogliere log, metadati, metriche e tracce. Puoi anche utilizzarlo ADOT per monitorare i servizi e impostare un allarme in base alle metriche inserite. CloudWatch

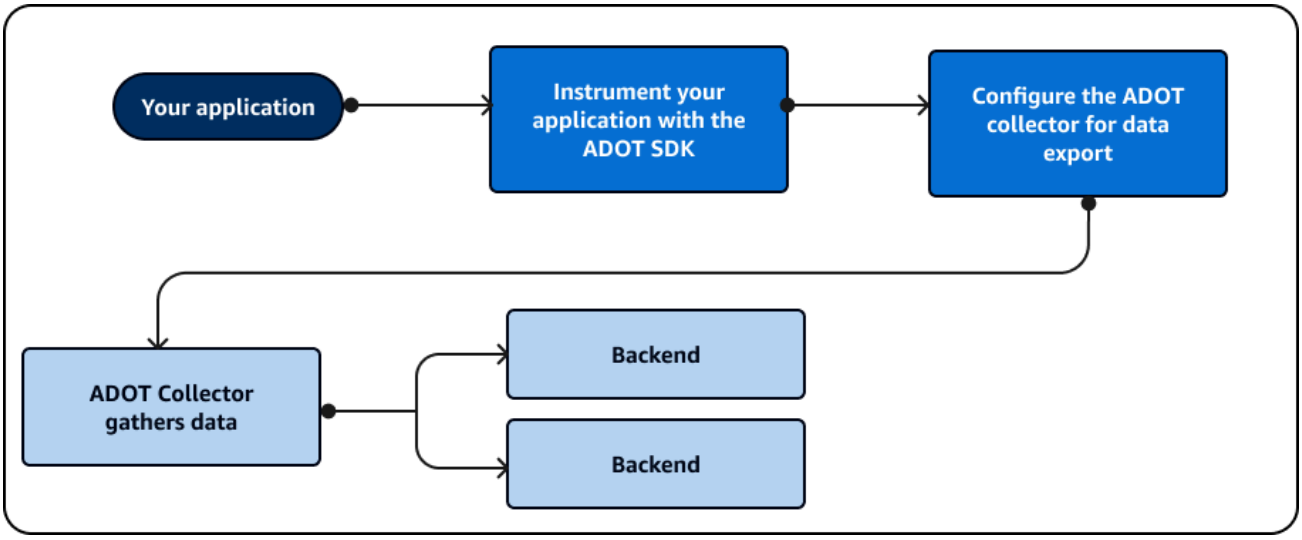
Se utilizzi l'ADOTSDK, hai le seguenti opzioni, in combinazione con un agente:

- Usa l'ADOTSDK con l'CloudWatch agente: consigliato.
- Usa l'ADOTSDK con ADOT Collector: consigliato se desideri utilizzare un software indipendente dal fornitore con AWS livelli di sicurezza e ottimizzazione.

Per utilizzare l'ADOTSDK, effettuate le seguenti operazioni:

- Strumenta la tua applicazione utilizzando l'ADOTSDK. Per ulteriori informazioni, consultate la documentazione relativa al vostro linguaggio di programmazione nella documentazione tecnica ADOT.
- Configura un ADOT raccoglitore per dirgli dove inviare i dati raccolti.

Dopo aver ricevuto i dati, il ADOT raccoglitore li invia al backend specificato nella configurazione. ADOT ADOTpuò inviare dati a più backend, inclusi fornitori esterni AWS, come mostrato nel diagramma seguente:



AWS si aggiorna regolarmente ADOT per aggiungere funzionalità e allinearsi al framework. OpenTelemetry Gli aggiornamenti e i piani futuri di sviluppo ADOT fanno parte di una tabella di marcia disponibile al pubblico. ADOTsupporta diversi linguaggi di programmazione, tra cui:

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

Se stai usando Python, ADOT puoi strumentare automaticamente la tua applicazione. Per iniziare a utilizzareADOT, consultate Introduzione e Guida introduttiva alla AWS Distro for OpenTelemetry Collector.

Usa il X-Ray SDK

L' X-Ray SDK è un set di AWS API e librerie che inviano dati ai servizi di AWS backend. Usa l' X-Ray SDK per strumentare la tua applicazione e raccogliere dati di tracciamento. Non è possibile utilizzare l' X-Ray SDK per raccogliere dati di registro o metrici.

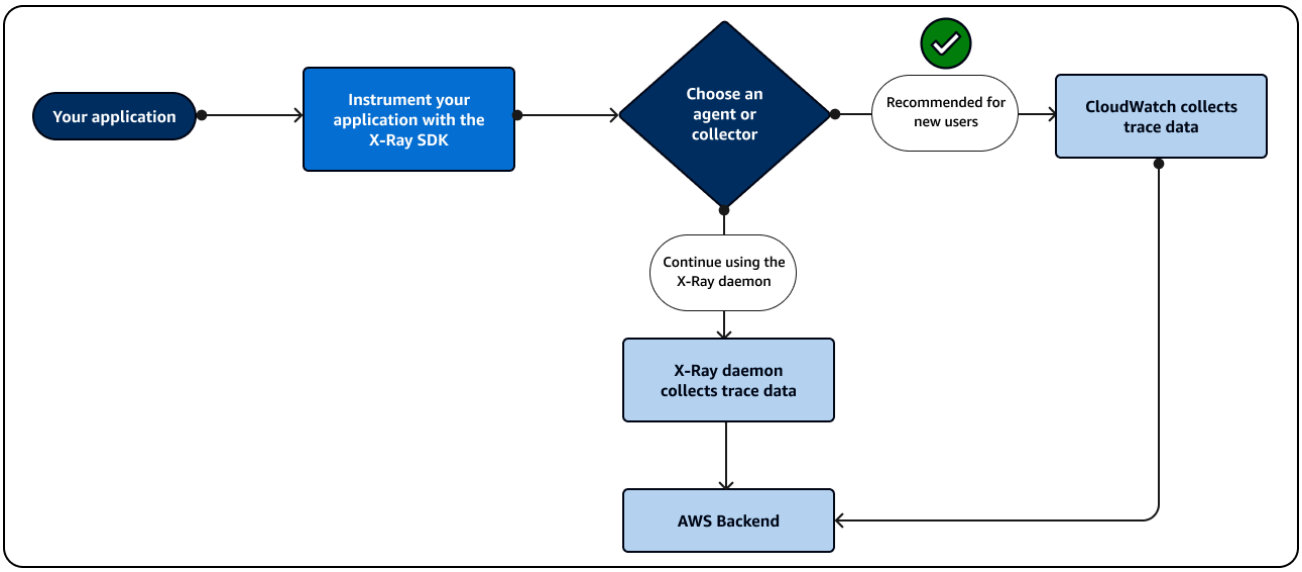
Se si utilizza l' X-Ray SDK, sono disponibili le seguenti opzioni, in combinazione con un agente:

- Usa l' X-Ray SDK conAWS X-Ray demone: usa questo se non vuoi aggiornare il codice dell'applicazione.
- Usa l' X-Ray SDK con l' CloudWatch agente: (consigliato) L' CloudWatch agente è compatibile con l' X-Ray SDK.

Per utilizzare l' X-Ray SDK, procedi come segue:

- Strumenta la tua applicazione utilizzando l' X-Ray SDK.
- Configura un raccoglitore per dirgli dove inviare i dati raccolti. È possibile utilizzare l' CloudWatch agente o il X-Ray demone per raccogliere le informazioni di tracciamento.

Dopo che il raccoglitore o l'agente riceve i dati, li invia a un AWS backend specificato nella configurazione dell'agente. L' X-Ray SDK può inviare dati a un AWS backend solo come mostrato nel diagramma seguente:



Se lo stai utilizzandoJava, puoi utilizzare l' X-Ray SDK per strumentare automaticamente la tua applicazione. Per iniziare a utilizzare l' X-Ray SDK, consulta le librerie associate ai seguenti linguaggi di programmazione:

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

- Ruby

Usa una console

Usa una console se desideri un'interfaccia utente grafica (GUI) che richieda una codifica minima. Gli utenti che non conoscono X-Ray possono iniziare rapidamente a utilizzare visualizzazioni predefinite ed eseguire attività di base. È possibile effettuare le seguenti operazioni direttamente dalla console:

- Abilita X-Ray.
- Visualizza riepiloghi di alto livello delle prestazioni della tua applicazione.
- Controlla lo stato di salute delle tue applicazioni.
- Identifica gli errori di alto livello.
- Visualizza i riepiloghi di base delle tracce.

Puoi utilizzare la CloudWatch console Amazon https://console.aws.amazon.com/cloudwatch/o la console X-Ray a https://console.aws.amazon.com/xray/casa per interagire con X-Ray.

Usa la CloudWatch console Amazon

La CloudWatch console include nuove funzionalità X-Ray che sono state riprogettate dalla console X-Ray per renderla più facile da usare. Se si utilizza la CloudWatch console, è possibile visualizzare CloudWatch log e metriche insieme ai dati di tracciamento X-Ray. Utilizza la CloudWatch console per visualizzare e analizzare i dati, tra cui:

- Tracce a raggi X: visualizza, analizza e filtra le tracce associate all'applicazione mentre soddisfa una richiesta. Utilizzate queste tracce per trovare latenze elevate, eseguire il debug degli errori e ottimizzare il flusso di lavoro delle applicazioni. Visualizza una mappa di tracciamento e una mappa dei servizi per visualizzare le rappresentazioni visive del flusso di lavoro dell'applicazione.
- Registri: visualizza, analizza e filtra i log prodotti dall'applicazione. Utilizza i log per risolvere gli errori e configurare il monitoraggio in base a valori di log specifici.
- Metriche: misura e monitora le prestazioni delle applicazioni utilizzando le metriche emesse dalle tue risorse o crea metriche personalizzate. Visualizza queste metriche in grafici e diagrammi.
- Monitoraggio delle reti e dell'infrastruttura: monitora le principali reti per rilevare eventuali interruzioni e lo stato e le prestazioni dell'infrastruttura, incluse applicazioni containerizzate, altri AWS servizi e client.

- Tutte le funzionalità della console X-Ray elencate nella seguente sezione Utilizzare la console X-Ray.

Per ulteriori informazioni sulla CloudWatch console, consulta la sezione Guida introduttiva ad Amazon CloudWatch.

Accedi alla CloudWatch console Amazon all'indirizzo https://console.aws.amazon.com/cloudwatch/.

Usa la console X-Ray

La console X-Ray offre la tracciabilità distribuita per le richieste delle applicazioni. Usa la console X-Ray se desideri un'esperienza di console più semplice o non desideri aggiornare il codice dell'applicazione. AWS non sta più sviluppando la console X-Ray. La console X-Ray contiene le seguenti funzionalità per le applicazioni strumentate:

- Approfondimenti: rileva automaticamente le anomalie nelle prestazioni dell'applicazione e individua le cause sottostanti. Gli approfondimenti sono inclusi nella CloudWatch console sotto Insights. Per ulteriori informazioni, vedere Use X-Ray Insights in. Usa la console X-Ray
- Mappa dei servizi: visualizza una struttura grafica dell'applicazione e le sue connessioni con client, risorse, servizi e dipendenze.
- Tracce: visualizza una panoramica delle tracce generate dall'applicazione mentre soddisfa una richiesta. Utilizza i dati di traccia per comprendere le prestazioni dell'applicazione rispetto a metriche di base, tra cui i tempi di HTTP risposta e risposta.
- Analisi: interpreta, esplora e analizza i dati di traccia utilizzando grafici per la distribuzione dei tempi di risposta.
- Configurazione: crea tracce personalizzate per modificare le configurazioni predefinite per quanto segue:
  - Campionamento: crea una regola che definisca la frequenza con cui campionare l'applicazione per le informazioni di traccia. Per ulteriori informazioni, consulta Configurare le regole di campionamento in. Usa la console X-Ray
  - Crittografia: crittografa i dati inattivi utilizzando una chiave che puoi controllare o disabilitare. AWS Key Management Service
  - Gruppi: utilizza un'espressione di filtro per definire un gruppo di tracce con una caratteristica comune come il nome di un URL o un tempo di risposta. Per ulteriori informazioni, consulta Configurare i gruppi.

Accedi alla console X-Ray da casahttps://console.aws.amazon.com/xray/.

Esplora la console X-Ray

Usa la console X-Ray per visualizzare una mappa dei servizi e delle tracce associate per le richieste servite dalle tue applicazioni e per configurare gruppi e regole di campionamento che influiscono sul modo in cui le tracce vengono inviate a X-Ray.

> **Note**
>
> La mappa e la mappa del servizio X-Ray sono state combinate nella CloudWatch ServiceLens mappa di tracciamento X-Ray all'interno della console Amazon. CloudWatch Apri la CloudWatchconsole e scegli Trace Map in X-Ray Traces dal riquadro di navigazione a sinistra.
> CloudWatch ora include Application Signals, che può scoprire e monitorare i servizi applicativi, i client, i canali Synthetics e le dipendenze dei servizi. Utilizza Application Signals per visualizzare un elenco o una mappa visiva dei tuoi servizi, visualizzare le metriche sanitarie in base agli obiettivi del livello di servizio (SLOs) e approfondire le tracce a raggi X correlate per una risoluzione dei problemi più dettagliata.

La pagina principale della console X-Ray è la mappa di traccia, che è una rappresentazione visiva del grafico del servizio JSON generato da X-Ray a partire dai dati di traccia generati dalle applicazioni. La mappa è composta da nodi di servizio per ogni applicazione che elabora richieste nel tuo account, nodi client a monte che rappresentano le origini delle richieste e nodi di servizio a valle che rappresentano i servizi e le risorse web utilizzate da un'applicazione durante l'elaborazione della richiesta. Sono disponibili pagine aggiuntive per la visualizzazione delle tracce e dei dettagli delle tracce e per la configurazione di gruppi e regole di campionamento.

Visualizza l'esperienza della console per X-Ray e confrontala con la CloudWatch console nelle sezioni seguenti.

Esplora X-Ray e le console CloudWatch

- Utilizzo della mappa di X-Ray tracciamento
- Visualizzazione delle tracce e dei dettagli delle tracce
- Utilizzo delle espressioni di filtro
- Cross-account tracciando

- Tracciamento delle applicazioni basate sugli eventi
- Utilizzo degli istogrammi di latenza
- Utilizzo degli X-Ray approfondimenti
- Interazione con la console di analisi
- Configurazione dei gruppi
- Configurazione delle regole di campionamento di
- Configurazione del campionamento adattivo
- Deep linking tra console

Utilizzo della mappa di X-Ray tracciamento

Visualizza la mappa di X-Ray tracciamento per identificare i servizi in cui si verificano errori, le connessioni con latenza elevata o le tracce delle richieste non riuscite.

> **Note**
>
> CloudWatch ora include Application Signals, in grado di rilevare e monitorare i servizi applicativi, i client, i canarini sintetici e le dipendenze dei servizi. Usa Application Signals per visualizzare un elenco o una mappa visiva dei tuoi servizi, visualizzare le metriche sullo stato di salute in base agli obiettivi del livello di servizio (SLO) e approfondire per visualizzare le tracce correlate X-Ray per una risoluzione più dettagliata dei problemi.
> La mappa e la mappa dei X-Ray servizi vengono combinate nella CloudWatch ServiceLens mappa di X-Ray tracciamento all'interno della console Amazon. CloudWatch Apri la CloudWatch console e scegli Trace Map sotto X-Ray Tracce dal pannello di navigazione a sinistra.

Visualizzazione della mappa di tracciamento

La mappa di traccia è una rappresentazione visiva dei dati di traccia generati dalle tue applicazioni. La mappa mostra i nodi di servizio che soddisfano le richieste, i nodi client a monte che rappresentano le origini delle richieste e i nodi di servizio a valle che rappresentano i servizi e le risorse Web utilizzati da un'applicazione durante l'elaborazione di una richiesta.

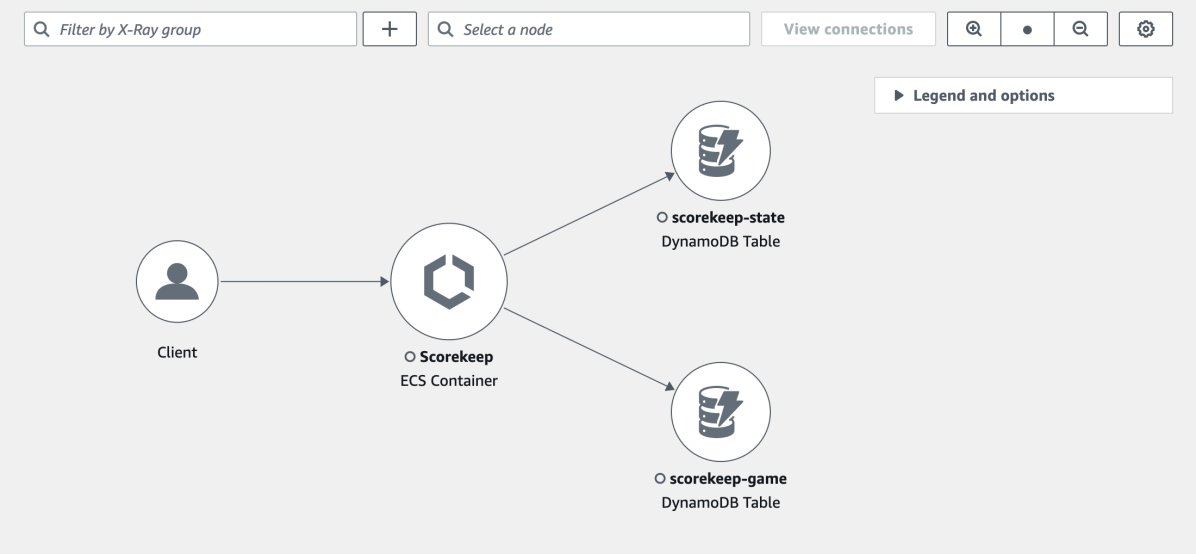
La mappa di tracciamento mostra una vista connessa delle tracce tra applicazioni basate sugli eventi che utilizzano Amazon SQS e Lambda. Per ulteriori informazioni, consulta Tracing event-driven

applications. La mappa di tracciamento supporta anche il tracciamento tra account, visualizzando i nodi di più account in un'unica mappa.

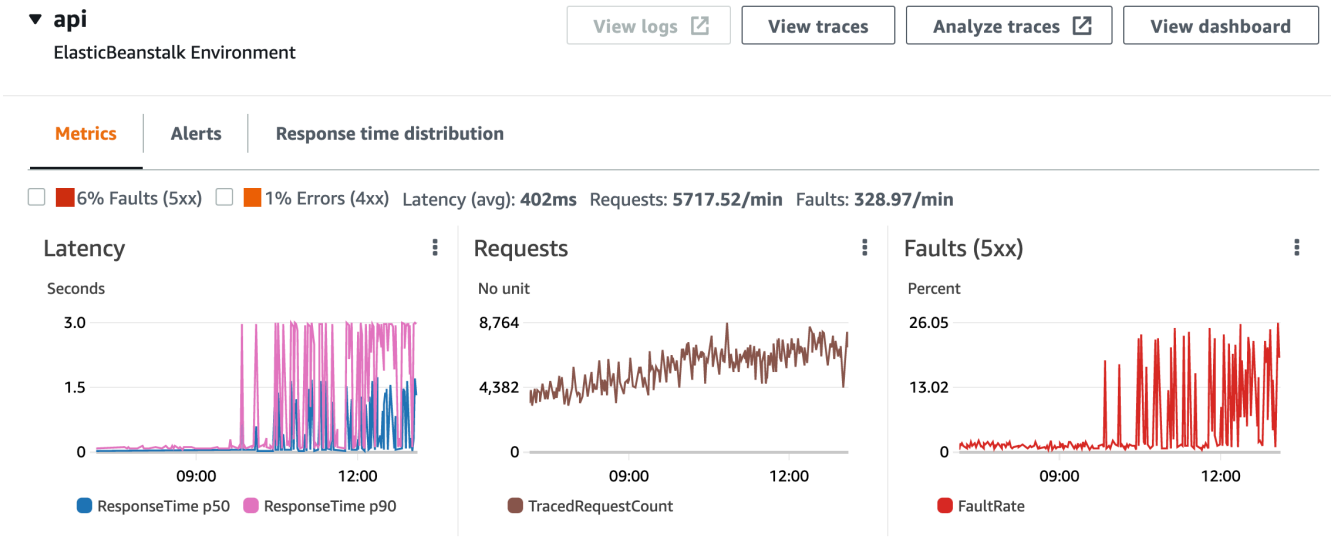
CloudWatch console

Per visualizzare la mappa di tracciamento nella console CloudWatch

1. Apri la CloudWatch console. Scegli Trace Map nella sezione X-Ray Tracce nel riquadro di navigazione a sinistra.



2. Scegliere un nodo di servizio per visualizzare le richieste per tale nodo o un collegamento tra due nodi per visualizzare le richieste che sono passate attraverso tale connessione.
3. Sotto la mappa di tracciamento vengono visualizzate informazioni aggiuntive, incluse le schede per le metriche, gli avvisi e la distribuzione dei tempi di risposta. Nella scheda Metriche, seleziona un intervallo all'interno di ciascun grafico per visualizzare maggiori dettagli, oppure scegli le opzioni Guasti o Errori per filtrare le tracce. Nella scheda Distribuzione del tempo di risposta, seleziona un intervallo all'interno del grafico per filtrare le tracce in base al tempo di risposta.



4. Visualizza le tracce scegliendo Visualizza tracce o, se è stato applicato un filtro, scegli Visualizza tracce filtrate.
5. Scegli Visualizza registri per visualizzare CloudWatch i registri associati al nodo selezionato. Non tutti i nodi della mappa di traccia supportano la visualizzazione dei log. Per ulteriori informazioni, consulta CloudWatch i log per la risoluzione dei problemi.

La mappa di tracciamento indica i problemi all'interno di ciascun nodo delineandolo con colori:

- Rosso per i malfunzionamenti del server (errori serie 500)
- Giallo per gli errori client (errori serie 400)
- Viola per gli errori di superamento dei limiti (429 Too Many Requests)

Se la tua mappa di tracciamento è grande, usa i controlli sullo schermo o il mouse per ingrandire e rimpicciolire e spostare la mappa.

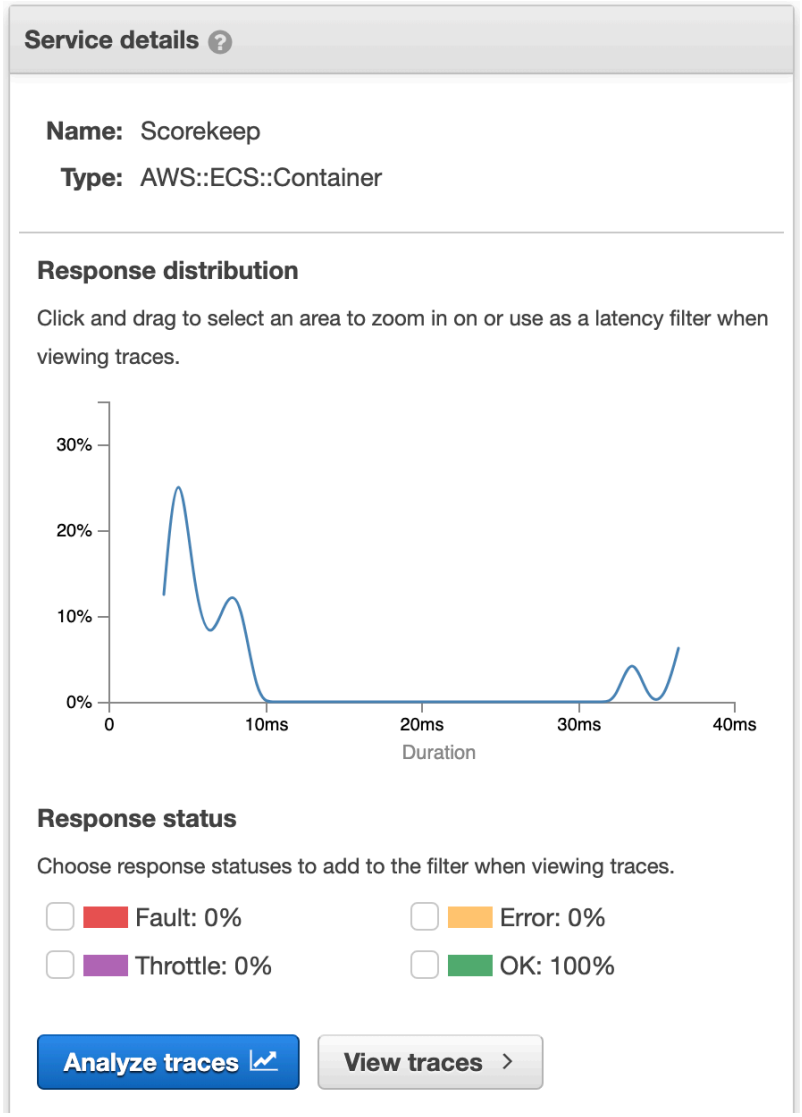
X-Ray console

Per visualizzare la mappa dei servizi

1. Apri la X-Ray console. La mappa dei servizi viene visualizzata per impostazione predefinita. Puoi anche scegliere Service Map dal riquadro di navigazione a sinistra.



2. Scegliere un nodo di servizio per visualizzare le richieste per tale nodo o un collegamento tra due nodi per visualizzare le richieste che sono passate attraverso tale connessione.
3. Utilizza l'istogramma di distribuzione delle risposte per filtrare le tracce in base alla durata e seleziona i codici di stato per i quali desideri visualizzare le tracce. Quindi scegliere View traces (Visualizza tracciamenti) per aprire l'elenco dei tracciamenti con l'espressione filtro applicata.



La mappa del servizio indica lo stato di salute di ogni nodo colorandolo in base al rapporto delle chiamate concluse con successo rispetto a errori e malfunzionamenti:

- Verde per le chiamate concluse con successo
- Rosso per i malfunzionamenti del server (errori serie 500)
- Giallo per gli errori client (errori serie 400)
- Viola per gli errori di superamento dei limiti (429 Too Many Requests)

Se la mappa del servizio è grande, usa i controlli sullo schermo o il mouse per ingrandire e rimpicciolire e spostare la mappa.

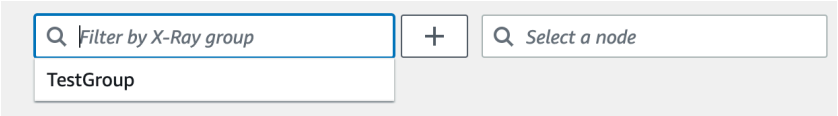
> **Note**
>
> La mappa di X-Ray tracciamento può visualizzare fino a 10.000 nodi. In rari scenari in cui il numero totale di nodi di servizio supera questo limite, potresti ricevere un errore e non essere in grado di visualizzare una mappa di traccia completa nella console.

Filtrare la mappa di tracciamento per gruppo

Utilizzando un'espressione di filtro, è possibile definire i criteri in base ai quali includere le tracce all'interno di un gruppo. Utilizzate i passaggi seguenti per visualizzare quindi quel gruppo specifico nella mappa di traccia.

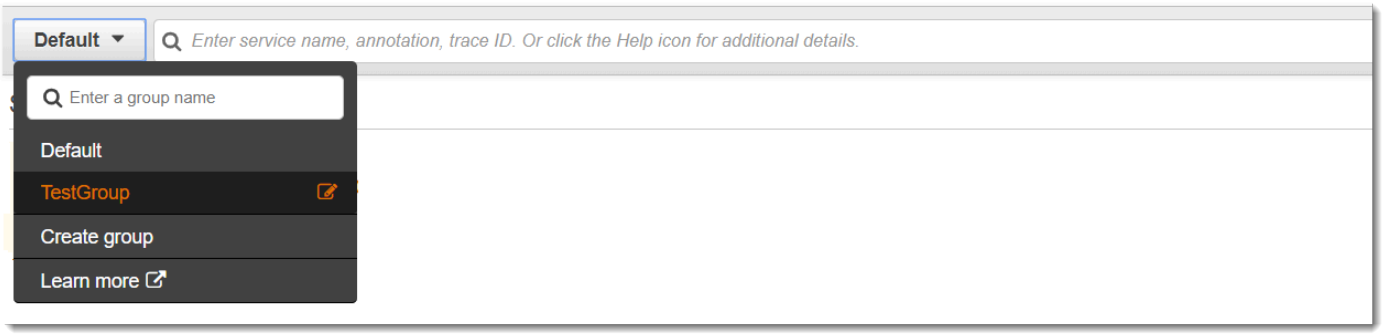
CloudWatch console

Scegli il nome di un gruppo dal filtro di gruppo in alto a sinistra della mappa di traccia.



X-Ray console

Scegliere un nome di gruppo dal menu a discesa a sinistra della barra di ricerca.



La mappa del servizio verrà ora filtrata per visualizzare le tracce che corrispondono all'espressione di filtro del gruppo selezionato.

Legenda e opzioni della mappa di traccia

La mappa di tracciamento include una legenda e diverse opzioni per personalizzare la visualizzazione della mappa.

CloudWatch console

Scegli il menu a discesa Legenda e opzioni in alto a destra sulla mappa. Scegli cosa visualizzare all'interno dei nodi, tra cui:

- Metrics mostra il tempo di risposta medio e il numero di tracce inviate al minuto nell'intervallo di tempo scelto.
- Nodes visualizza l'icona del servizio all'interno di ciascun nodo.

Scegli le impostazioni aggiuntive della mappa dal pannello Preferenze, a cui puoi accedere tramite l'icona a forma di ingranaggio in alto a destra della mappa. Queste impostazioni includono la selezione della metrica da utilizzare per determinare la dimensione di ciascun nodo e quali canarini devono essere visualizzati sulla mappa.

X-Ray console

Visualizza la legenda della mappa dei servizi scegliendo il link Legenda della mappa in alto a destra sulla mappa. Le opzioni della mappa dei servizi possono essere scelte nella parte inferiore destra della mappa di tracciamento, tra cui:

- Service Icons attiva e disattiva la visualizzazione all'interno di ciascun nodo, visualizzando l'icona del servizio o il tempo di risposta medio e il numero di tracce inviate al minuto durante l'intervallo di tempo scelto.
- Dimensionamento dei nodi: Nessuno imposta tutti i nodi alla stessa dimensione.
- Dimensionamento dei nodi: Health dimensiona i nodi in base al numero di richieste interessate, inclusi errori, guasti o richieste limitate.
- Dimensionamento dei nodi: il traffico dimensiona i nodi in base al numero totale di richieste.

Visualizzazione delle tracce e dei dettagli delle tracce

Usa la pagina Tracce nella X-Ray console per trovare le tracce tramite URL, codice di risposta o altri dati dal riepilogo della traccia. Dopo aver selezionato una traccia dall'elenco delle tracce, la pagina dei dettagli della traccia mostra una mappa dei nodi di servizio associati alla traccia selezionata e una cronologia dei segmenti di traccia.

Visualizzazione dei tracciamenti

CloudWatch console

Per visualizzare le tracce nella console CloudWatch

1. Accedi Console di gestione AWS e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel riquadro di navigazione a sinistra, scegli X-Ray Tracce, quindi scegli Tracce. Puoi filtrare per gruppo o inserire un'espressione di filtro. Questo filtra le tracce visualizzate nella sezione Tracce nella parte inferiore della pagina.

   In alternativa, è possibile utilizzare la mappa dei servizi per accedere a un nodo di servizio specifico e quindi visualizzare le tracce. Si apre la pagina Traces con una query già applicata.
3. Perfeziona la tua query nella sezione Query Refiners. Per filtrare le tracce in base a un attributo comune, scegli un'opzione dalla freccia in giù accanto a Perfeziona la query per. Le opzioni sono le seguenti:

   - Nodo: filtra le tracce per nodo di servizio.
   - Resource ARN: filtra le tracce in base a una risorsa associata a una traccia. Esempi di queste risorse includono un'istanza Amazon Elastic Compute Cloud (Amazon EC2), una AWS Lambda funzione o una tabella. Amazon DynamoDB
   - Utente: filtra le tracce con un ID utente.
   - Messaggio relativo alla causa principale dell'errore: filtra le tracce in base alla causa principale dell'errore.
   - URL: filtra le tracce in base al percorso URL utilizzato dall'applicazione.
   - Codice di stato HTTP: filtra le tracce in base al codice di stato HTTP restituito dall'applicazione. È possibile specificare un codice di risposta personalizzato o selezionare una delle seguenti opzioni:
     - `200`— La richiesta è andata a buon fine.
     - `401`— Nella richiesta mancavano credenziali di autenticazione valide.
     - `403`— La richiesta non disponeva di autorizzazioni valide.
     - `404`— Il server non è riuscito a trovare la risorsa richiesta.
     - `500`— Il server ha riscontrato una condizione imprevista e ha generato un errore interno.

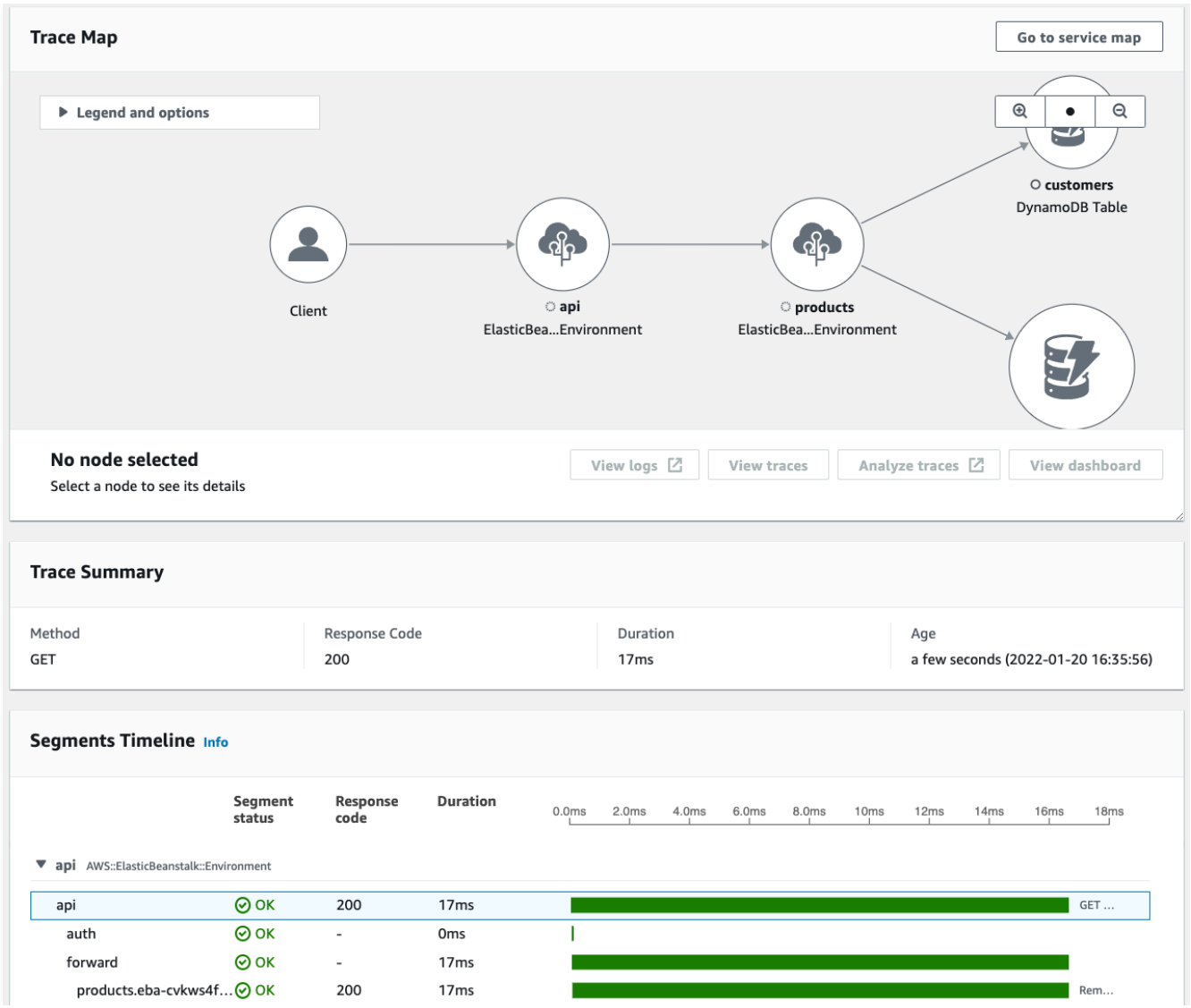
Scegli una o più voci, quindi scegli Aggiungi alla query per aggiungerle all'espressione di filtro nella parte superiore della pagina.

4. Per trovare una singola traccia, inserisci un ID di traccia direttamente nel campo della query. Puoi utilizzare il X-Ray formato o il formato World Wide Web Consortium (W3C). Ad esempio, una traccia creata utilizzando il AWS Distro for OpenTelemetry è in formato W3C.

   > **Note**
   >
   > Quando si interrogano le tracce create con un ID di W3C-format traccia, la console visualizza la traccia corrispondente nel formato. X-Ray Ad esempio, se si esegue una query `4efaaf4d1e8720b39541901950019ee5` in formato W3C, la console visualizza l' X-Ray equivalente:. `1-4efaaf4d-1e8720b39541901950019ee5`

5. Scegli Esegui la query in qualsiasi momento per visualizzare un elenco di tracce corrispondenti nella sezione Tracce nella parte inferiore della pagina.
6. Per visualizzare la pagina dei dettagli della traccia per una singola traccia, seleziona un ID di traccia dall'elenco.

   L'immagine seguente mostra una mappa di traccia contenente i nodi di servizio associati alla traccia e i bordi tra i nodi che rappresentano il percorso intrapreso dai segmenti che compongono la traccia. Un riepilogo di Trace segue la Trace Map. Il riepilogo contiene informazioni su un'`GET`operazione di esempio, il relativo codice di risposta, la durata impiegata per l'esecuzione della traccia e l'età della richiesta. La cronologia dei segmenti segue il riepilogo della traccia che mostra la durata dei segmenti e dei sottosegmenti di traccia.



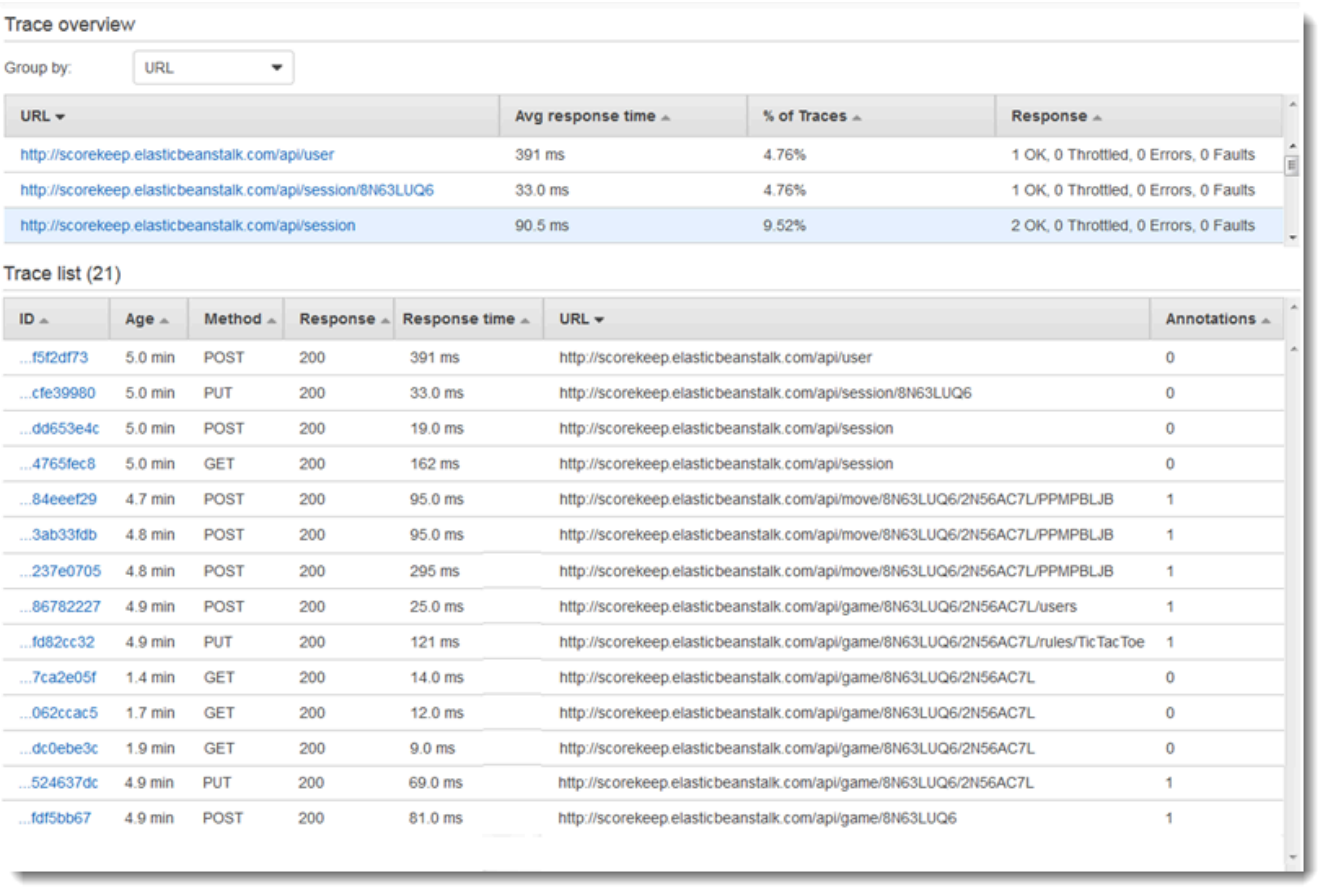
Se disponi di un'applicazione basata sugli eventi che utilizza Amazon SQS e Lambda, puoi visualizzare una vista connessa delle tracce per ogni richiesta nella mappa Trace. Nella mappa, le tracce dei produttori di messaggi sono collegate alle tracce dei AWS Lambda consumatori e vengono visualizzate come un bordo tratteggiato. Per ulteriori informazioni sulle applicazioni basate sugli eventi, vedere. Tracciamento delle applicazioni basate sugli eventi

Le pagine dei dettagli di Traces e Trace supportano anche il tracciamento tra account, che può elencare le tracce di più account nell'elenco di tracce e all'interno di una singola mappa di traccia.

X-Ray console

Per visualizzare le tracce nella console X-Ray

1. Apri la pagina Tracce nella X-Ray console. Il pannello di panoramica delle tracce mostra un elenco di tracce raggruppate per caratteristiche comuni, tra cui Error root causes, ResourceARN e. InstanceId
2. Per selezionare una feature comune per visualizzare un insieme raggruppato di tracce, espandi la freccia verso il basso accanto a Raggruppa per. L'illustrazione seguente mostra una panoramica delle tracce raggruppate per URL e un elenco delle tracce associate. AWS X-Ray applicazione di esempio



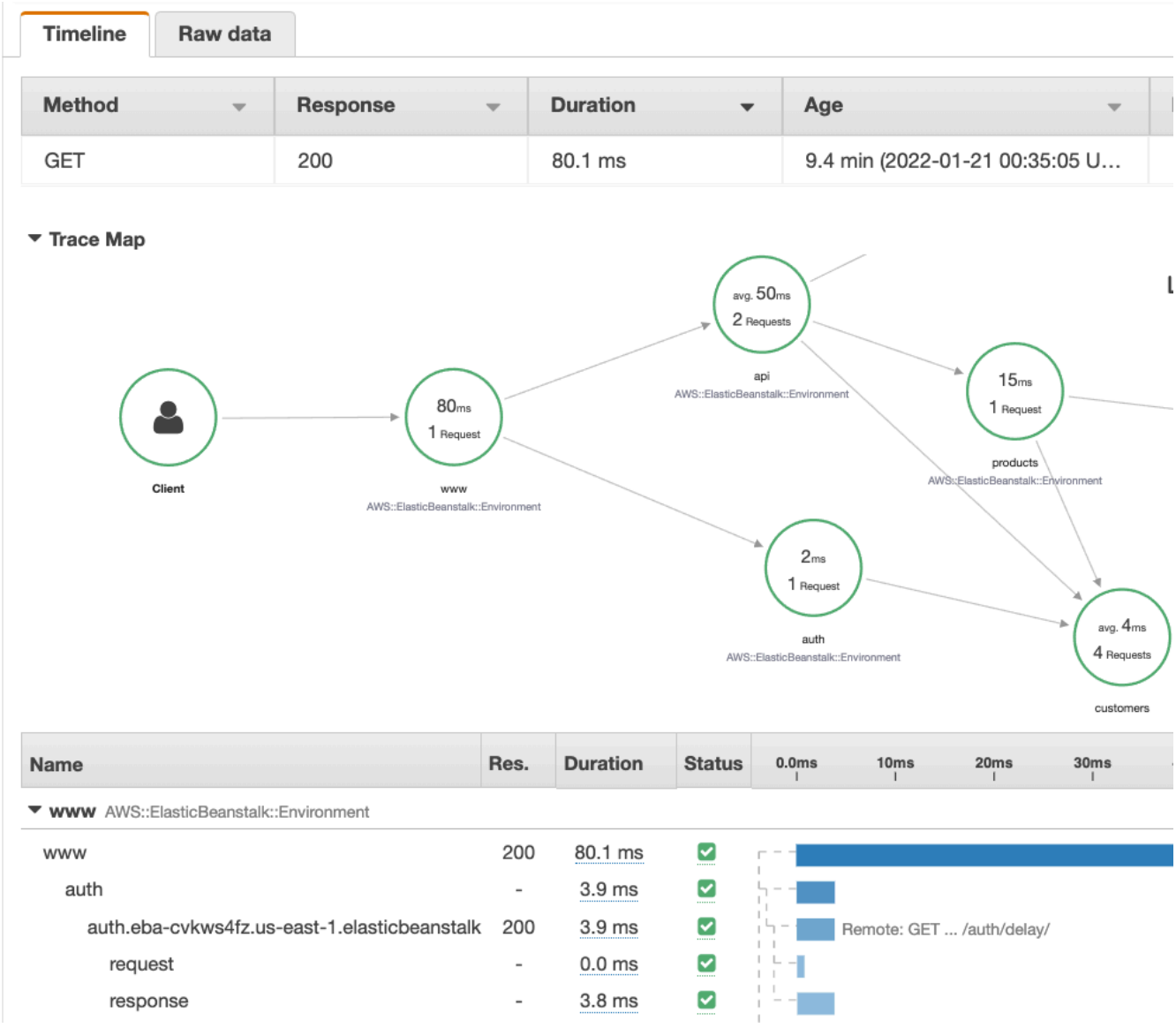
3. Scegli l'ID di una traccia per visualizzarla nell'elenco Trace. Puoi anche scegliere Service map nel pannello di navigazione per visualizzare le tracce per un nodo di servizio specifico. Quindi puoi visualizzare le tracce associate a quel nodo.

   La scheda Timeline mostra il flusso di richieste per la traccia e include quanto segue:

   - Una mappa del percorso per ogni segmento della traccia.

- Quanto tempo ha impiegato il segmento per raggiungere un nodo nella mappa di tracciamento.
- Quante richieste sono state fatte al nodo nella mappa di tracciamento.

La figura seguente mostra un esempio di Trace Map associata a una `GET` richiesta effettuata a un'applicazione di esempio. Le frecce mostrano il percorso seguito da ciascun segmento per completare la richiesta. I nodi del servizio mostrano il numero di richieste effettuate durante la `GET` richiesta.



Per ulteriori informazioni sulla scheda Timeline, consulta la seguente sezione Esplorazione della timeline di traccia.

La scheda Dati grezzi mostra le informazioni sulla traccia e sui segmenti e sottosegmenti che la compongono, in formato. JSON Queste informazioni possono includere quanto segue:

- Timestamp
- ID univoci
- Risorse associate al segmento o sottosegmento
- L'origine o l'origine del segmento o sottosegmento
- Informazioni aggiuntive sulla richiesta all'applicazione, ad esempio la risposta a una richiesta HTTP

Esplorazione della cronologia di tracciamento

La sezione Cronologia mostra una gerarchia di segmenti e sottosegmenti accanto a una barra orizzontale che corrisponde al tempo impiegato per completare le attività. La prima voce nell'elenco è il segmento, che rappresenta tutti i dati registrati tramite il servizio per una singola richiesta. I sottosegmenti sono rientrati ed elencati dopo il segmento. Le colonne contengono informazioni su ciascun segmento.

CloudWatch console

Nella CloudWatch console, la timeline dei segmenti fornisce le seguenti informazioni:

- La prima colonna: elenca i segmenti e i sottosegmenti nella traccia selezionata.
- La colonna Stato del segmento: elenca il risultato dello stato di ogni segmento e sottosegmento.
- La colonna Codice di risposta: elenca un codice di stato della risposta HTTP a una richiesta del browser effettuata dal segmento o sottosegmento, se disponibile.
- La colonna Durata: elenca la durata del segmento o sottosegmento.
- La colonna Ospitato in: elenca lo spazio dei nomi o l'ambiente in cui viene eseguito il segmento o sottosegmento, se applicabile. Per ulteriori informazioni, consulta Dimensioni raccolte e combinazioni di dimensioni.
- L'ultima colonna: visualizza le barre orizzontali che corrispondono alla durata di funzionamento del segmento o sottosegmento, in relazione agli altri segmenti o sottosegmenti nella timeline.

Per raggruppare l'elenco di segmenti e sottosegmenti per nodo di servizio, attiva Raggruppa per nodi.

X-Ray console

Nella pagina dei dettagli della traccia, scegli la scheda Cronologia per visualizzare la cronologia di ogni segmento e sottosegmento che compone una traccia.

Nella X-Ray console, la timeline fornisce le seguenti informazioni:

- La colonna Nome: elenca i nomi dei segmenti e dei sottosegmenti nella traccia.
- La colonna Res.: elenca un codice di stato della risposta HTTP a una richiesta del browser effettuata dal segmento o sottosegmento, se disponibile.
- La colonna Durata: elenca la durata del segmento o sottosegmento.
- La colonna Stato: elenca il risultato dello stato del segmento o sottosegmento.
- Ultima colonna: visualizza le barre orizzontali che corrispondono alla durata di funzionamento del segmento o sottosegmento, in relazione agli altri segmenti o sottosegmenti nella timeline.

Per visualizzare i dati di traccia non elaborati utilizzati dalla console per generare la timeline, scegli la scheda Dati grezzi. I dati grezzi mostrano informazioni sulla traccia e sui segmenti e sottosegmenti che la compongono in formato. JSON Queste informazioni possono includere quanto segue:

- Timestamp
- ID univoci
- Risorse associate al segmento o sottosegmento
- L'origine o l'origine del segmento o sottosegmento
- Informazioni aggiuntive sulla richiesta all'applicazione, ad esempio la risposta a una richiesta HTTP.

Quando si utilizza un AWS SDK o un SQL client strumentato per effettuare chiamate a risorse esterne, l' X-Ray SDK registra automaticamente i sottosegmenti. HTTP Puoi anche utilizzare l' X-Ray SDK per registrare sottosegmenti personalizzati per qualsiasi funzione o blocco di codice. I sottosegmenti aggiuntivi registrati mentre un sottosegmento personalizzato è aperto diventano figli del sottosegmento personalizzato.

Visualizzazione dei dettagli del segmento

Dalla timeline di tracciamento, scegliete il nome di un segmento per visualizzarne i dettagli.

Il pannello dei dettagli del segmento mostra le schede Panoramica, Risorse, Annotazioni, Metadati, Eccezioni e SQL. Si applica quanto segue:

- La scheda Overview (Panoramica) mostra le informazioni su richiesta e risposta. Le informazioni includono il nome, l'ora di inizio, l'ora di fine, la durata, l'URL della richiesta, l'operazione della richiesta, il codice di risposta alla richiesta ed eventuali errori e guasti.
- La scheda Risorse per un segmento mostra le informazioni dall' X-RaySDK e sulle AWS risorse che eseguono l'applicazione. Usa i plug-in Amazon EC2 o Amazon ECS per l' X-Ray SDK per registrare informazioni sulle risorse specifiche del servizio. AWS Elastic Beanstalk Per ulteriori informazioni sui plug-in, consulta la sezione Plugin di servizio in. Configurazione dell'X-Ray SDK per Java
- Le schede rimanenti mostrano le annotazioni, i metadati e le eccezioni registrati per il segmento. Le eccezioni vengono acquisite automaticamente quando vengono generate da una richiesta strumentata. Le annotazioni e i metadati contengono informazioni aggiuntive che vengono registrate utilizzando le operazioni fornite dall'SDK. X-Ray Per aggiungere annotazioni o metadati ai segmenti, usa l'SDK. X-Ray Per ulteriori informazioni, consulta il link specifico della lingua elencato in Strumenting your application with SDK in. AWS X-Ray Strumentazione della tua applicazione per AWS X-Ray

Visualizzazione dei dettagli del sottosegmento

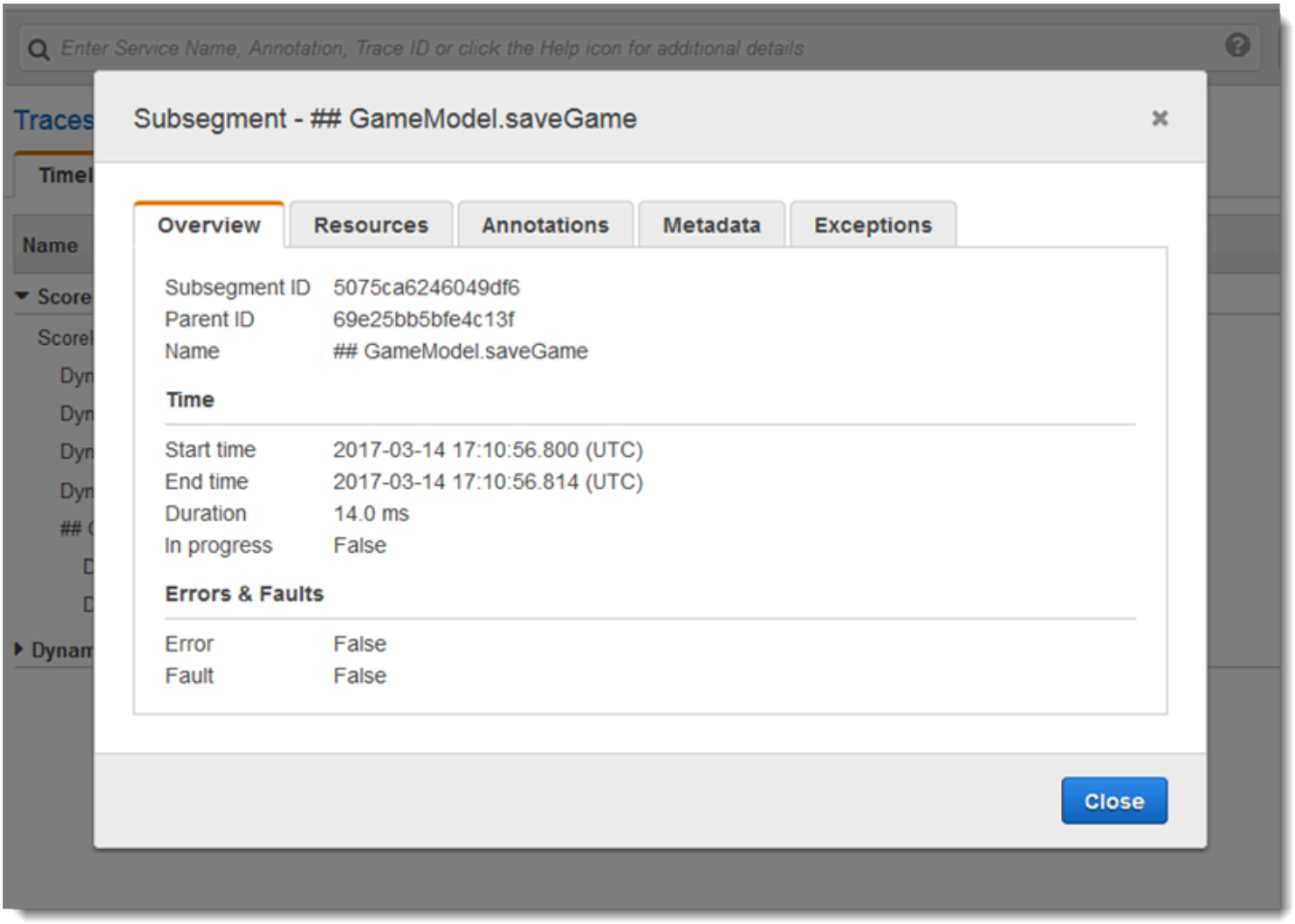
Dalla timeline di tracciamento, scegli il nome di un sottosegmento per visualizzarne i dettagli:

- La scheda Panoramica contiene informazioni sulla richiesta e sulla risposta. Ciò include il nome, l'ora di inizio, l'ora di fine, la durata, la richiestaURL, l'operazione della richiesta, il codice di risposta alla richiesta ed eventuali errori e guasti. Per i sottosegmenti generati dai client analizzati, la scheda Overview (Panoramica) contiene informazioni sulla richiesta e la risposta dal punto di vista dell'applicazione.
- La scheda Risorse per un sottosegmento mostra i dettagli sulle AWS risorse utilizzate per eseguire il sottosegmento. Ad esempio, la scheda delle risorse può includere una AWS Lambda funzione ARN, informazioni su una tabella DynamoDB, qualsiasi operazione chiamata e l'ID della richiesta.

- Le schede rimanenti mostrano le annotazioni, i metadati e le eccezioni registrati nel sottosegmento. Le eccezioni vengono acquisite automaticamente quando vengono generate da una richiesta strumentata. Le annotazioni e i metadati contengono informazioni aggiuntive che vengono registrate utilizzando le operazioni fornite dall'SDK. X-Ray Usa l' X-Ray SDK per aggiungere annotazioni o metadati ai tuoi segmenti. Per ulteriori informazioni, consulta il link specifico della lingua elencato in Strumenting your application with SDK in. AWS X-Ray Strumentazione della tua applicazione per AWS X-Ray

Nel caso di sottosegmenti personalizzati, la scheda Overview (Panoramica) mostra il nome del sottosegmento, che è possibile impostare per specificare l'area del codice o della funzione che memorizza. Per ulteriori informazioni, consulta il link specifico della lingua elencato in Strumentazione dell'applicazione con SDK in. AWS X-Ray Generazione di sottosegmenti personalizzati con l' X-Ray SDK for Java

L'immagine seguente mostra la scheda Panoramica per un sottosegmento personalizzato. La panoramica contiene l'ID del sottosegmento, l'ID principale, il nome, l'ora di inizio e di fine, la durata, lo stato e gli errori o i guasti.



La scheda Metadati per un sottosegmento personalizzato contiene informazioni in JSON formato sulle risorse utilizzate da quel sottosegmento.

Utilizzo delle espressioni di filtro

Utilizza le espressioni di filtro per visualizzare una o più tracce per una richiesta, un servizio, una connessione tra due servizi (un edge) o richieste che soddisfano una condizione. X-Ray fornisce un linguaggio di espressione di filtro per filtrare richieste, servizi e bordi in base ai dati nelle intestazioni delle richieste, allo stato della risposta e ai campi indicizzati sui segmenti originali.

Quando scegli un periodo di tempo per le tracce da visualizzare nella X-Ray console, potresti ottenere più risultati di quelli che la console può visualizzare. Nell'angolo in alto a destra, la console mostra il numero di tracciamenti individuati e se sono disponibili ulteriori tracciamenti. È possibile utilizzare un'espressione di filtro per restringere i risultati alle sole tracce che si desidera trovare.

Argomenti

- Filtraggio dei dettagli delle espressioni
- Utilizzo delle espressioni filtro con i gruppi
- Sintassi delle espressioni filtro
- Parole chiave booleane
- Parole chiave numeriche
- Parole chiave stringa
- Parole chiave complesse
- funzione id

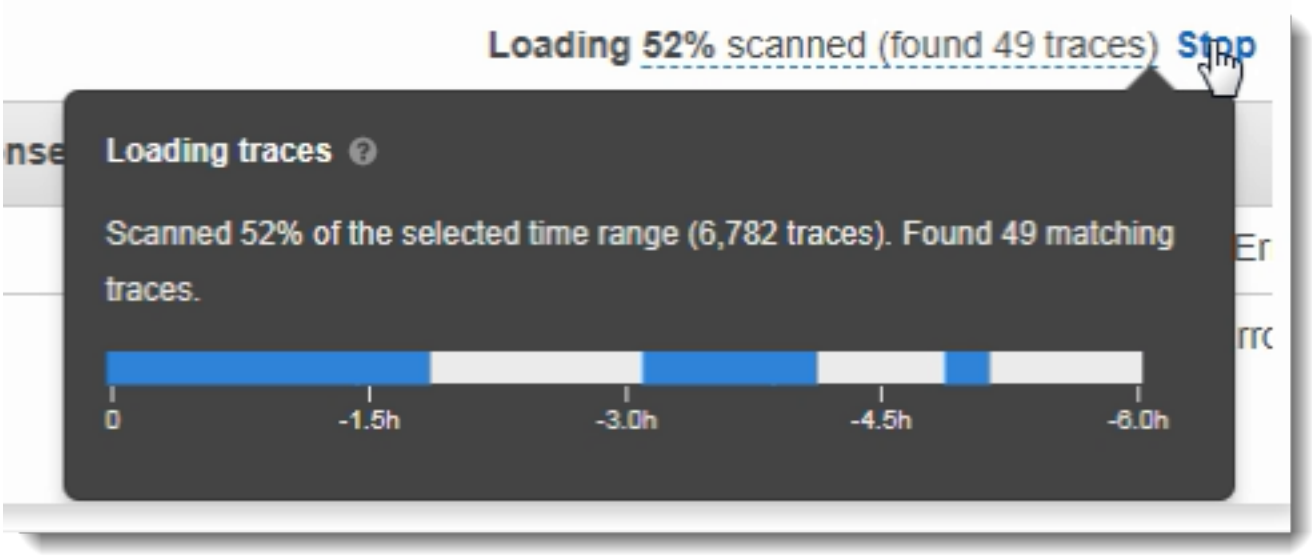
Filtraggio dei dettagli delle espressioni

Quando si sceglie un nodo nella mappa di traccia, la console crea un'espressione di filtro in base al nome di servizio del nodo e ai tipi di errore presenti in base alla selezione effettuata. Per individuare tracciamenti che mostrano problemi di prestazioni o che riguardano richieste specifiche, puoi modificare l'espressione generata dalla console oppure crearne una tua. Se aggiungi annotazioni con l' X-Ray SDK, puoi anche filtrare in base alla presenza di una chiave di annotazione o al valore di una chiave.

> **Note**
>
> Se scegli un intervallo di tempo relativo nella mappa di traccia e scegli un nodo, la console converte l'intervallo di tempo in un'ora di inizio e di fine assolute. Per garantire che i tracciamenti relativi al nodo vengano visualizzati tra risultati di ricerca, e per evitare la scansione dei periodi di tempo in cui il nodo non era attivo, l'intervallo di tempo include solo i periodi durante i quali il nodo ha inviato dei tracciamenti. Se desideri eseguire una ricerca rispetto all'ora corrente, puoi ripristinare l'intervallo di tempo relativo nella pagina dei tracciamenti ed eseguire nuovamente la scansione.

Se vengono ancora restituiti più risultati di quanti la console possa visualizzare, la console mostra quanti tracciamenti soddisfano il filtro e di quanti è stata eseguita la scansione. La percentuale visualizzata è la percentuale dell'intervallo di tempo selezionato di cui è stata eseguita la scansione. Restringi ulteriormente l'espressione filtro, oppure scegli un intervallo di tempo più breve, in modo da visualizzare tra i risultati tutti i tracciamenti corrispondenti al filtro.

Per disporre prima dei risultati più recenti, la console avvia la scansione partendo dal termine dell'intervallo di tempo e procedendo in senso cronologico inverso. In caso di un numero elevato di tracciamenti, la console suddivide l'intervallo di tempo in blocchi ed esegue le scansioni in parallelo. La barra di avanzamento mostra le porzioni dell'intervallo di tempo relativamente alle quali è stata eseguita la scansione.



Utilizzo delle espressioni filtro con i gruppi

I gruppi costituiscono una raccolta di tracce definite da un'espressione di filtro. Puoi utilizzare i gruppi per generare grafici di servizio aggiuntivi e fornire CloudWatch metriche Amazon.

I gruppi sono identificati dal nome o da un Amazon Resource Name (ARN) e contengono un'espressione di filtro. Il servizio confronta i tracciamenti in entrata con l'espressione e le memorizza di conseguenza.

È possibile creare e modificare i gruppi utilizzando il menu a discesa a sinistra della barra di ricerca delle espressioni filtro.

> **Note**
>
> Se il servizio rileva un errore nel qualificare un gruppo, tale gruppo non viene più incluso nell'elaborazione dei tracciamenti in entrata e viene registrata un parametro di errore.

Per ulteriori informazioni sui gruppi, consulta. Configurazione dei gruppi

Sintassi delle espressioni filtro

Le espressioni filtro possono contenere una parola chiave, un operatore unario o binario e un valore per il confronto.

```
keyword operator value
```

Sono disponibili diversi operatori per diversi tipi di parola chiave. Ad esempio, `responsetime` è una parola chiave numerica e può essere confrontata con operatori correlati a numeri.

Example— richieste in cui il tempo di risposta è stato superiore a 5 secondi

```
responsetime > 5
```

Puoi combinare più espressioni in un'espressione composta mediante gli operatori `AND` o `OR`.

Example— richieste la cui durata totale era di 5—8 secondi

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

Le parole chiave e gli operatori semplici individuano problemi solo a livello di tracciamento. Se si verifica un errore a valle, ma è gestito direttamente dall'applicazione e non viene restituito all'utente, una ricerca per `error` non lo individuerà.

Per individuare tracciamenti che includano problemi con chiamate a valle, puoi utilizzare le parole chiave complesse `service()` e `edge()`. Queste parole chiave ti permettono di applicare un'espressione filtro a tutti i nodi a valle, a un singolo nodo a valle o ad un collegamento tra due nodi. Per una maggiore granularità, puoi filtrare servizi ed edge per tipo con la funzione id().

Parole chiave booleane

I valori delle parole chiave booleane sono true o false. Utilizza queste parole chiave per individuare i tracciamenti che hanno generato errori.

Parole chiave booleane

- ok— Il codice di stato della risposta era 2XX Success.
- `error`— Il codice di stato della risposta era 4XX Client Error.

- `throttle`— Il codice di stato della risposta era 429 Too Many Requests.
- `fault`— Il codice di stato della risposta era 5XX Server Error.
- `partial`— La richiesta ha segmenti incompleti.
- `inferred`— La richiesta ha segmenti dedotti.
- `first`— L'elemento è il primo di un elenco enumerato.
- `last`— L'elemento è l'ultimo di un elenco enumerato.
- `remote`— L'entità della causa principale è remota.
- `root`— Il servizio è il punto di ingresso o il segmento principale di una traccia.

Gli operatori booleani individuano i segmenti in cui la chiave specificata è `true` o `false`.

Operatori booleani

- nessuno: l'espressione è vera se la parola chiave è vera.
- `!`— L'espressione è vera se la parola chiave è falsa.
- =, `!=` — Confronta il valore della parola chiave con la stringa `true` or`false`. Questi operatori agiscono allo stesso modo degli altri operatori, ma sono più espliciti.

Example— lo stato della risposta è 2XX OK

```
ok
```

Example— lo stato della risposta non è 2XX OK

```
!ok
```

Example— lo stato della risposta non è 2XX OK

```
ok = false
```

Example— l'ultima traccia di errore enumerata ha il nome di errore «deserializza»

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

Example— richieste con segmenti remoti in cui la copertura è maggiore di 0,7 e il nome del servizio è «trace»

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

Example— richieste con segmenti dedotti in cui il tipo di servizio è "» AWS:DynamoDB

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

Example— richieste che hanno come radice un segmento con il nome «data-plane»

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

Parole chiave numeriche

Utilizzare le parole chiave numeriche per ricercare le richieste con un tempo di risposta, una durata o un codice di stato della risposta specifici.

Parole chiave numeriche

- `responsetime`— Tempo impiegato dal server per inviare una risposta.
- `duration`— Durata totale della richiesta, incluse tutte le chiamate downstream.
- `http.status`— Codice di stato della risposta.
- `index`— Posizione di un elemento in un elenco enumerato.
- `coverage`— Percentuale decimale del tempo di risposta dell'entità rispetto al tempo di risposta del segmento principale. Applicabile solo per le entità causa principale del tempo di risposta.

Operatori numerici

Le parole chiave numeriche utilizzano gli operatori di uguaglianza e di confronto standard.

- =, `!` = — La parola chiave è uguale o diversa da un valore numerico.
- <,<=,>, >= — La parola chiave è minore o maggiore di un valore numerico.

Example— lo stato della risposta non è 200 OK

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

Example— richiesta in cui la durata totale era di 5—8 secondi

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

Example— richieste completate con successo in meno di 3 secondi, comprese tutte le chiamate a valle

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

Example— entità di elenco enumerato con un indice maggiore di 5

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

Example— richieste in cui l'ultima entità con copertura è superiore a 0,8

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

Parole chiave stringa

Le parole chiave stringa permettono di trovare i tracciamenti con testo specifico all'interno delle intestazioni della richiesta o con specifici ID utente.

Parole chiave stringa

- `http.url`— URL della richiesta.
- `http.method`— Metodo di richiesta.
- `http.useragent`— Richiedere la stringa dell'agente utente.
- `http.clientip`— Indirizzo IP del richiedente.
- `user`— Valore del campo utente su qualsiasi segmento della traccia.
- `name`— Il nome di un servizio o di un'eccezione.
- `type`— Tipo di servizio.
- `message`— Messaggio di eccezione.
- `availabilityzone`— Valore del campo availabilityzone su qualsiasi segmento della traccia.
- `instance.id`— Valore del campo ID dell'istanza su qualsiasi segmento della traccia.
- `resource.arn`— Valore del campo ARN della risorsa su qualsiasi segmento della traccia.

Gli operatori di stringa selezionano valori che sono uguali a, o contengono, un testo specifico. I valori devono essere sempre specificati tra virgolette.

Operatori di stringa

- =, != — La parola chiave è uguale o diversa da un valore numerico.
- `CONTAINS`— La parola chiave contiene una stringa specifica.
- `BEGINSWITH`, `ENDSWITH` — La parola chiave inizia o termina con una stringa specifica.

Example— filtro http.url

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

Per verificare se un campo esiste all'interno di un tracciamento, indipendentemente dal suo valore, verificare se contiene la stringa vuota.

Example— filtro utente

Trova tutti i tracciamenti contenenti ID degli utenti.

```
user CONTAINS ""
```

Example— seleziona le tracce con una causa principale dell'errore che include un servizio denominato «Auth»

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

Example— seleziona le tracce con una root cause del tempo di risposta il cui ultimo servizio ha un tipo di DynamoDB

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

Example— seleziona le tracce con una causa principale di errore la cui ultima eccezione è il messaggio «accesso negato per account_id: 1234567890"

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

Parole chiave complesse

Le parole chiave complesse permettono di selezionare le richieste in base a nome del servizio, nome edge o valore dell'annotazione. Per servizi e collegamenti, puoi specificare un ulteriore espressione filtro che si applica al servizio o al collegamento. Per le annotazioni, puoi filtrare in base al valore di un'annotazione con una specifica chiave utilizzando operatori booleani, numerici o stringa.

Parole chiave complesse

- `annotation[`*`key`*`]`— Valore di un'annotazione con campo. *`key`* Il valore di un'annotazione può essere un valore booleano, numerico o stringa, perciò puoi usare un operatore di confronto di uno qualsiasi di questi tipi. È possibile utilizzare questa parola chiave in combinazione con la `edge` parola chiave `service` o. Una chiave di annotazione che contiene punti (punti) deve essere racchiusa tra parentesi quadre ([]).
- `edge(`*`source`*`, `*`destination`*`) {`*`filter`*`}`— Connessione tra servizi e. *`source`* *`destination`* Le parentesi graffe opzionali possono contenere un'espressione filtro che si applica ai segmenti della connessione.
- `group.`*`name`* `/ group.`*`arn`*— Il valore dell'espressione di filtro di un gruppo, a cui fa riferimento il nome del gruppo o l'ARN del gruppo.
- `json`— Oggetto della causa principale JSON. Vedi Acquisizione di dati da AWS X-Ray per informazioni sulla procedura per creare entità JSON a livello di codice.
- `service(`*`name`*`) {`*`filter`*`}`— Servizio con nome. *`name`* Le parentesi graffe opzionali possono contenere un'espressione filtro che si applica ai segmenti creati dal servizio.

Utilizza la parola chiave service per trovare le tracce delle richieste che raggiungono un determinato nodo sulla mappa di tracciamento.

Gli operatori di parole chiave complesse trovano i segmenti in cui è stata impostata o non è stata impostata la chiave specificata.

Operatori di parole chiave complessi

- nessuno: l'espressione è vera se la parola chiave è impostata. Se la parola chiave è di tipo booleano, restituirà il valore booleano.
- `!`— L'espressione è vera se la parola chiave non è impostata. Se la parola chiave è di tipo booleano, restituirà il valore booleano.
- `=`, `!=` — Confronta il valore della parola chiave.

- `edge(`*`source`*`, `*`destination`*`) {`*`filter`*`}`— Connessione tra servizi *`source`* e*`destination`*. Le parentesi graffe opzionali possono contenere un'espressione filtro che si applica ai segmenti della connessione.
- `annotation[`*`key`*`]`— Valore di un'annotazione con campo*`key`*. Il valore di un'annotazione può essere un valore booleano, numerico o stringa, perciò puoi usare un operatore di confronto di uno qualsiasi di questi tipi. È possibile utilizzare questa parola chiave in combinazione con la `edge` parola chiave `service` o.
- `json`— Oggetto della causa principale JSON. Vedi Acquisizione di dati da AWS X-Ray per informazioni sulla procedura per creare entità JSON a livello di codice.

Utilizza la parola chiave service per trovare le tracce delle richieste che raggiungono un determinato nodo sulla mappa di tracciamento.

Example— Filtro di servizio

Richieste che includevano una chiamata a `api.example.com` e che si sono concluse con un malfunzionamento (errore serie 500).

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

Puoi escludere il nome del servizio per applicare un'espressione filtro a tutti i nodi della tua mappa del servizio.

Example— filtro di servizio

Richieste che hanno causato un errore in qualsiasi punto della mappa di tracciamento.

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

La parola chiave edge applica un'espressione filtro a una connessione tra due nodi.

Example— filtro perimetrale

Richiesta in cui il servizio `api.example.com` ha eseguito una chiamata verso `backend.example.com` che si è conclusa con un errore.

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

Puoi inoltre utilizzare l'operatore `!` con le parole chiave service ed edge per escludere un servizio o un collegamento dai risultati di un'altra espressione filtro.

Example— filtro di servizio e richiesta

Richiesta in cui l'URL inizia con `http://api.example.com/` e contiene `/v2/` ma non raggiunge un servizio denominato `api.example.com`.

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

Example— filtro del servizio e del tempo di risposta

Trova le tracce dove `http url` è impostato e il tempo di risposta è superiore a 2 secondi.

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

Per le annotazioni, puoi richiamare tutte le tracce in cui `annotation[`*`key`*`]` è impostato o utilizzare gli operatori di confronto che corrispondono al tipo di valore.

Example— annotazione con valore di stringa

Richieste con un'annotazione denominata `gameid` con valore stringa `"817DL6VO"`.

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

Example— l'annotazione è impostata

Richieste con un'annotazione denominata `age` set.

```
annotation[age]
```

Example— l'annotazione non è impostata

Richieste senza un'annotazione denominata `age` set.

```
!annotation[age]
```

Example— annotazione con valore numerico

Richieste con annotazione age con valore numerico maggiore di 29.

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

Example— annotazione in combinazione con service o edge

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

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

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

Example— gruppo con utente

Richiede in cui le tracce soddisfano il filtro di `high_response_time` gruppo (ad esempio`responseTime > 3`) e l'utente si chiama Alice.

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

Example— JSON con entità root cause

Le richieste con entità causa principale corrispondenti.

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

funzione id

Quando indichi un nome di servizio per le parole chiave `service` o `edge`, ottieni risultati per tutti i nodi con tale nome. Per una maggiore precisione del filtraggio, puoi utilizzare la funzione `id` per specificare un tipo di servizio in aggiunta ad un nome per distinguere tra nodi con lo stesso nome.

Utilizza la `account.id` funzione per specificare un particolare account per il servizio, quando visualizzi le tracce di più account in un account di monitoraggio.

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

Puoi utilizzare la funzione `id` al posto di un nome di servizio nei filtri service ed edge.

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

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

Ad esempio, AWS Lambda le funzioni generano due nodi nella mappa di traccia, uno per la chiamata della funzione e uno per il servizio Lambda. I due nodi hanno lo stesso nome ma appartengono a tipi diversi. Un filtro sul servizio standard troverà i tracciamenti relativi a entrambi.

Example— filtro di servizio

Richieste che includono un errore su qualsiasi servizio denominato `random-name`.

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

Utilizza la funzione `id` per restringere la ricerca a errori sulla funzione stessa, escludendo gli errori del servizio.

Example— filtro di servizio con funzione id

Richieste che includono un errore su un servizio denominato `random-name` di tipo `AWS::Lambda::Function`.

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

Per ricercare i nodi in base al tipo, puoi anche omettere completamente il nome.

Example— filtro di servizio con funzione id e tipo di servizio

Richieste che includono un errore su un servizio di tipo `AWS::Lambda::Function`.

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

Per cercare i nodi per un particolare Account AWS, specifica un ID di account.

Example— filtro di servizio con funzione id e ID account

Richieste che includono un servizio all'interno di un ID account specifico`AWS::Lambda::Function`.

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

Cross-account tracciando

AWS X-Ray supporta l'osservabilità tra più account, consentendo di monitorare e risolvere i problemi delle applicazioni che si estendono su più account all'interno di un. Regione AWS Puoi cercare, visualizzare e analizzare senza problemi le tue metriche, i log e le tracce in qualsiasi account collegato come se operassi in un unico account. Ciò fornisce una visione completa delle richieste che viaggiano su più account. È possibile visualizzare le tracce tra account nella mappa di X-Ray tracciamento e nelle pagine delle tracce all'interno della CloudWatch console.

I dati di osservabilità condivisi possono includere uno dei seguenti tipi di telemetria:

- Metriche in Amazon CloudWatch
- Gruppi di log in Amazon CloudWatch Logs
- Tracce in AWS X-Ray
- Applicazioni in Amazon CloudWatch Application Insights

Configura l'osservabilità tra più account

Per attivare l'osservabilità tra più account, configura uno o più account di AWS monitoraggio e collegali a più account di origine. Un account di monitoraggio è una centrale in Account AWS grado di visualizzare e interagire con i dati di osservabilità generati dagli account di origine. Un account di origine è un individuo Account AWS che genera dati di osservabilità per le risorse che contiene.

Gli account di origine condividono i dati di osservabilità con gli account di monitoraggio. Le tracce vengono copiate da ciascun account di origine su un massimo di cinque account di monitoraggio. Le copie delle tracce dagli account di origine al primo account di monitoraggio sono gratuite. Le copie delle tracce inviate ad altri account di monitoraggio vengono addebitate su ciascun account di origine, in base ai prezzi standard. Per ulteriori informazioni, consulta AWS X-Ray i prezzi e i CloudWatch prezzi di Amazon.

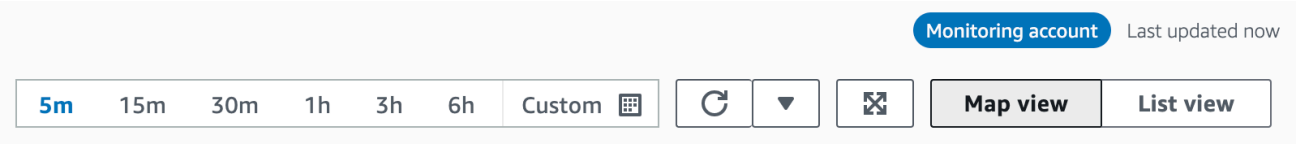
Per creare collegamenti tra gli account di monitoraggio e gli account di origine, utilizza la CloudWatch console o i nuovi comandi Observability Access Manager nell'API AWS CLI and. Per maggiori informazioni, consulta la sezione Osservabilità su più account di CloudWatch .

> **Note**
>
> X-Ray le tracce vengono fatturate nel Account AWS luogo in cui vengono ricevute. Se una richiesta di esempio riguarda più servizi Account AWS, ogni account registra una traccia

separata e tutte le tracce condividono lo stesso ID di traccia. Per saperne di più sui prezzi osservabili su più account, consulta i prezzi e i AWS X-Ray prezzi di Amazon. CloudWatch

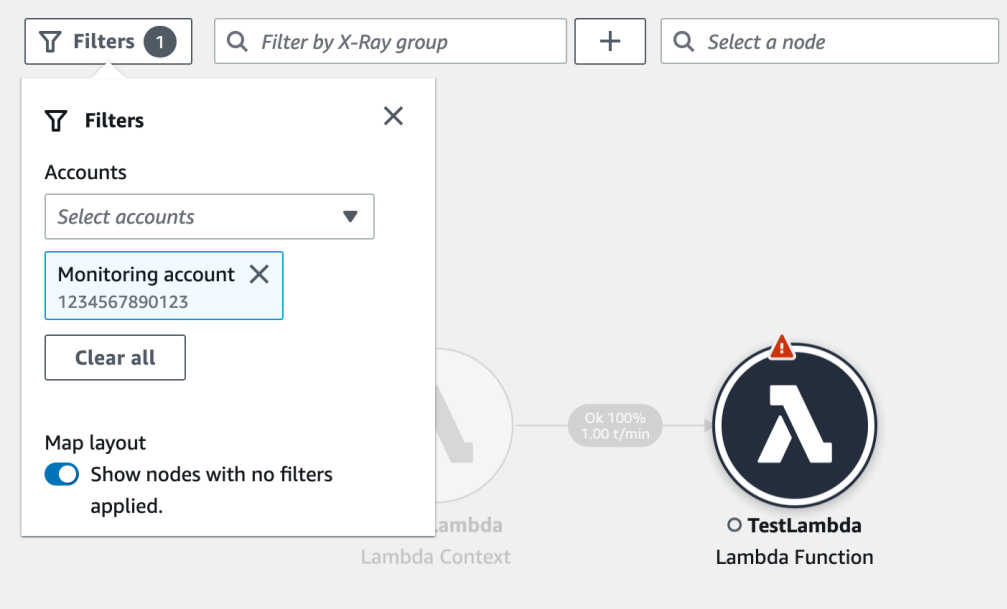
Visualizzazione delle tracce tra più account

Cross-account le tracce vengono visualizzate nell'account di monitoraggio. Ogni account di origine mostra solo le tracce locali per quell'account specifico. Le sezioni seguenti presuppongono che tu abbia effettuato l'accesso all'account di monitoraggio e abbia aperto la CloudWatch console Amazon. Sia nella pagina della mappa di tracciamento che nella pagina delle tracce, nell'angolo in alto a destra viene visualizzato un badge dell'account di monitoraggio.

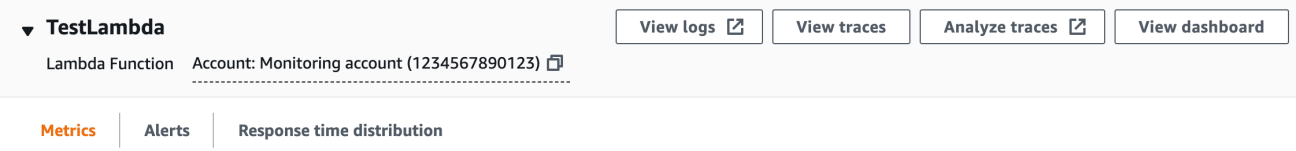


Mappa delle tracce

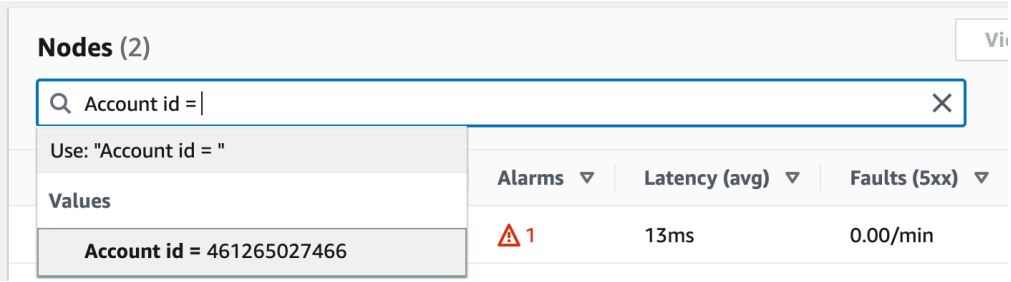
Nella CloudWatch console, scegli Trace Map sotto X-Ray Tracce dal riquadro di navigazione a sinistra. Per impostazione predefinita, la mappa di tracciamento mostra i nodi per tutti gli account di origine che inviano tracce all'account di monitoraggio e i nodi per l'account di monitoraggio stesso. Sulla mappa di traccia, scegli Filtri in alto a sinistra per filtrare la mappa di traccia utilizzando il menu a discesa Account. Dopo l'applicazione di un filtro sull'account, i nodi di servizio degli account che non corrispondono al filtro corrente vengono visualizzati in grigio.



Quando si sceglie un nodo di servizio, il riquadro dei dettagli del nodo include l'ID e l'etichetta dell'account del servizio.



Nell'angolo in alto a destra della mappa di tracciamento, scegli Visualizzazione elenco per visualizzare un elenco di nodi di servizio. L'elenco dei nodi di servizio include i servizi dell'account di monitoraggio e tutti gli account di origine configurati. Filtra l'elenco dei nodi in base all'etichetta dell'account o all'ID dell'account scegliendoli dal filtro Nodi.

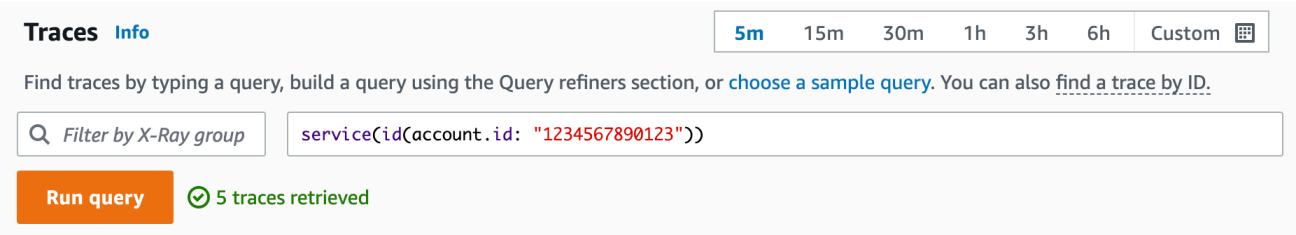


Tracce

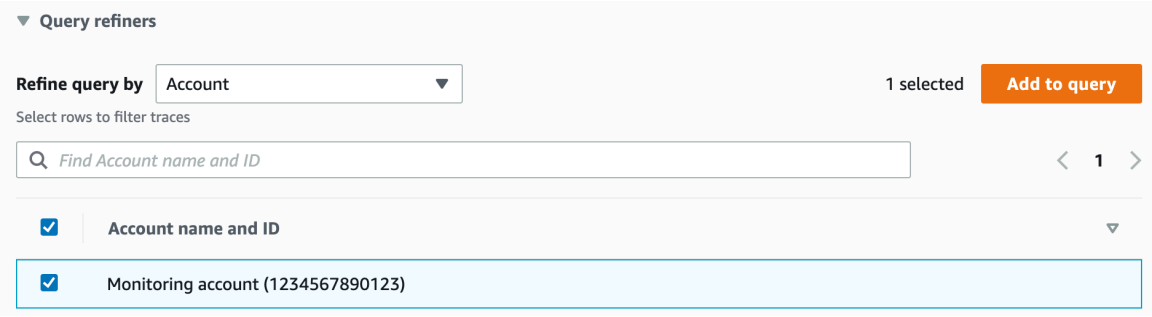
Visualizza i dettagli delle tracce relative a più account aprendo la CloudWatch console dall'account di monitoraggio e scegliendo Tracce sotto Tracce nel X-Ray riquadro di navigazione a sinistra. Puoi anche aprire questa pagina scegliendo un nodo nella X-Ray Trace Map, quindi scegliendo Visualizza tracce dal riquadro dei dettagli del nodo.

La pagina Traces supporta l'interrogazione in base all'ID dell'account. Per iniziare, inserisci una query che includa uno o più ID account. L'esempio seguente richiede le tracce che sono passate attraverso l'ID dell'account X o Y:

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



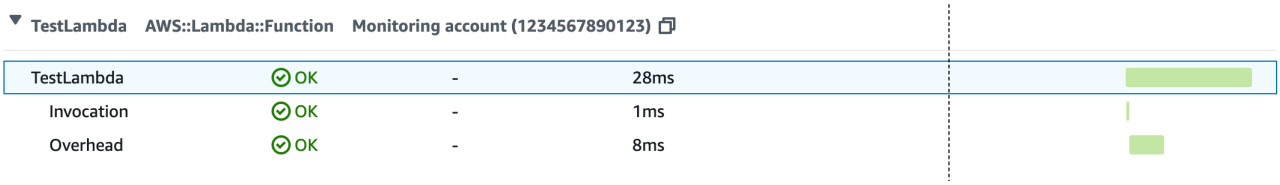
Perfeziona la tua richiesta per Account. Seleziona uno o più account dall'elenco e scegli Aggiungi alla query.



Dettagli della traccia

Visualizza i dettagli di una traccia scegliendola dall'elenco Tracce nella parte inferiore della pagina Tracce. Vengono visualizzati i dettagli della traccia, inclusa una mappa dei dettagli della traccia con i nodi di servizio di tutti gli account attraversati dalla traccia. Scegli un nodo di servizio specifico per visualizzare l'account corrispondente.

La sezione Cronologia dei segmenti mostra i dettagli dell'account per ogni segmento della timeline.



Tracciamento delle applicazioni basate sugli eventi

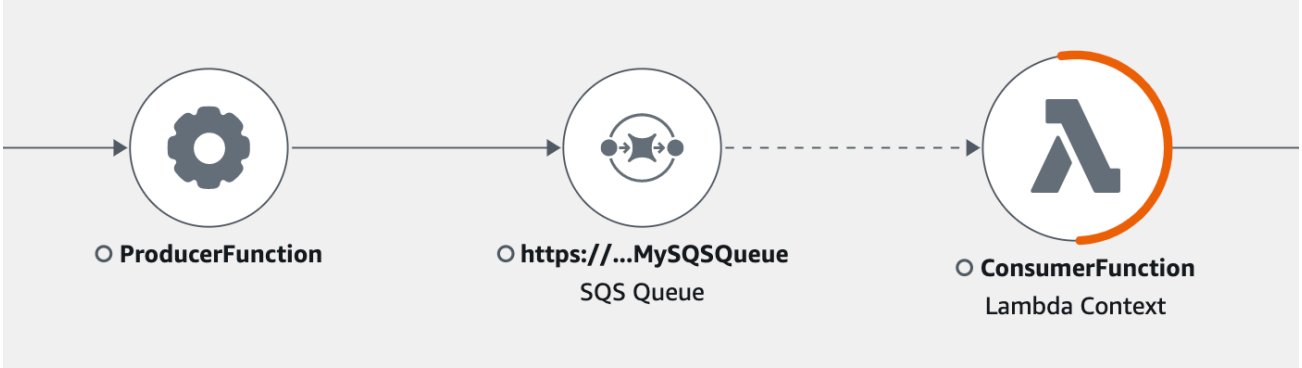
AWS X-Ray supporta il tracciamento di applicazioni basate sugli eventi utilizzando Amazon SQS e. AWS Lambda Usa la CloudWatch console per visualizzare una vista connessa di ogni richiesta mentre viene accodata con Amazon SQS ed elaborata da una o più funzioni Lambda. Le tracce provenienti dai produttori di messaggi upstream vengono automaticamente collegate alle tracce provenienti dai nodi consumer Lambda a valle, creando una vista end-to-end dell'applicazione.

> **Note**
>
> Ogni segmento di traccia può essere collegato a un massimo di 20 tracce, mentre una traccia può includere un massimo di 100 collegamenti. In alcuni scenari, il collegamento di tracce aggiuntive può comportare il superamento della dimensione massima del documento di traccia, provocando una traccia potenzialmente incompleta. Ciò può accadere, ad esempio, quando una funzione Lambda con la traccia abilitata invia molti messaggi SQS a una coda con una singola chiamata. Se riscontri questo problema, è disponibile una soluzione che

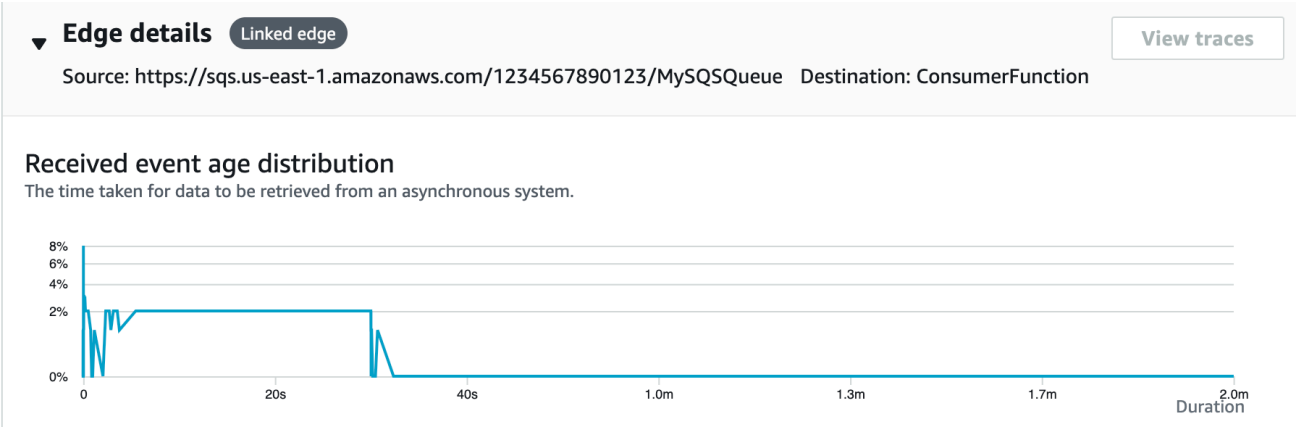
utilizza gli SDK. X-Ray Per ulteriori informazioni, consulta l' X-Ray SDK per Java Node.js, Python , Go o .NET.

Visualizza le tracce collegate nella mappa delle tracce

Usa la pagina Trace Map all'interno della CloudWatch console per visualizzare una mappa di traccia con le tracce dei produttori di messaggi collegate alle tracce dei consumatori Lambda. Questi collegamenti sono visualizzati con un bordo tratteggiato che collega il nodo Amazon SQS e i nodi consumer Lambda a valle.



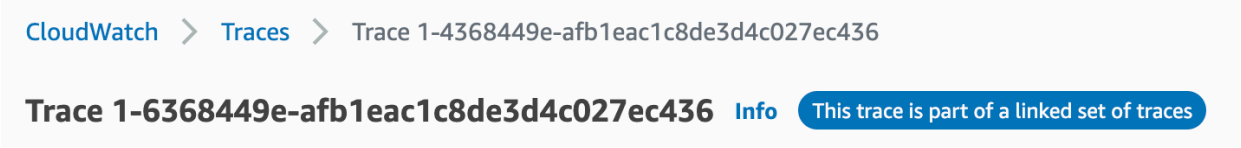
Seleziona un bordo tratteggiato per visualizzare un istogramma dell'età dell'evento ricevuto, che mappa la diffusione dell'età dell'evento quando viene ricevuto dai consumatori. L'età viene calcolata ogni volta che viene ricevuto un evento.



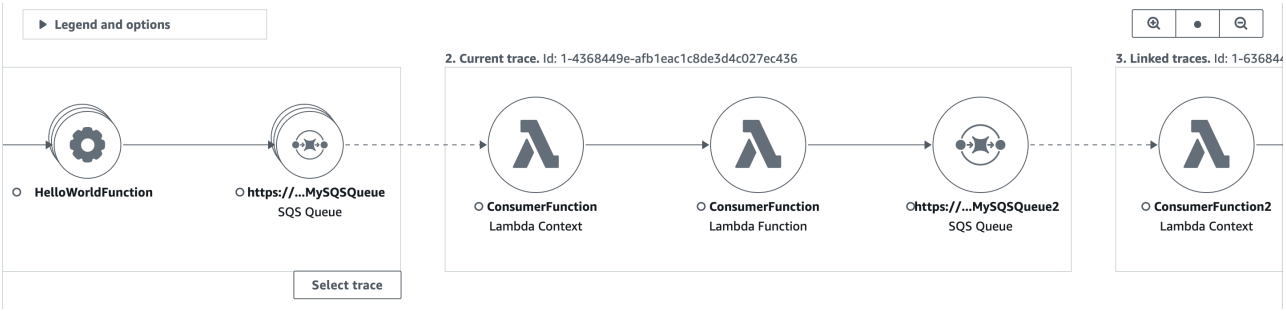
Visualizza i dettagli della traccia collegata

Visualizza i dettagli di tracciamento inviati da un produttore di messaggi, da una coda Amazon SQS o da un consumatore Lambda:

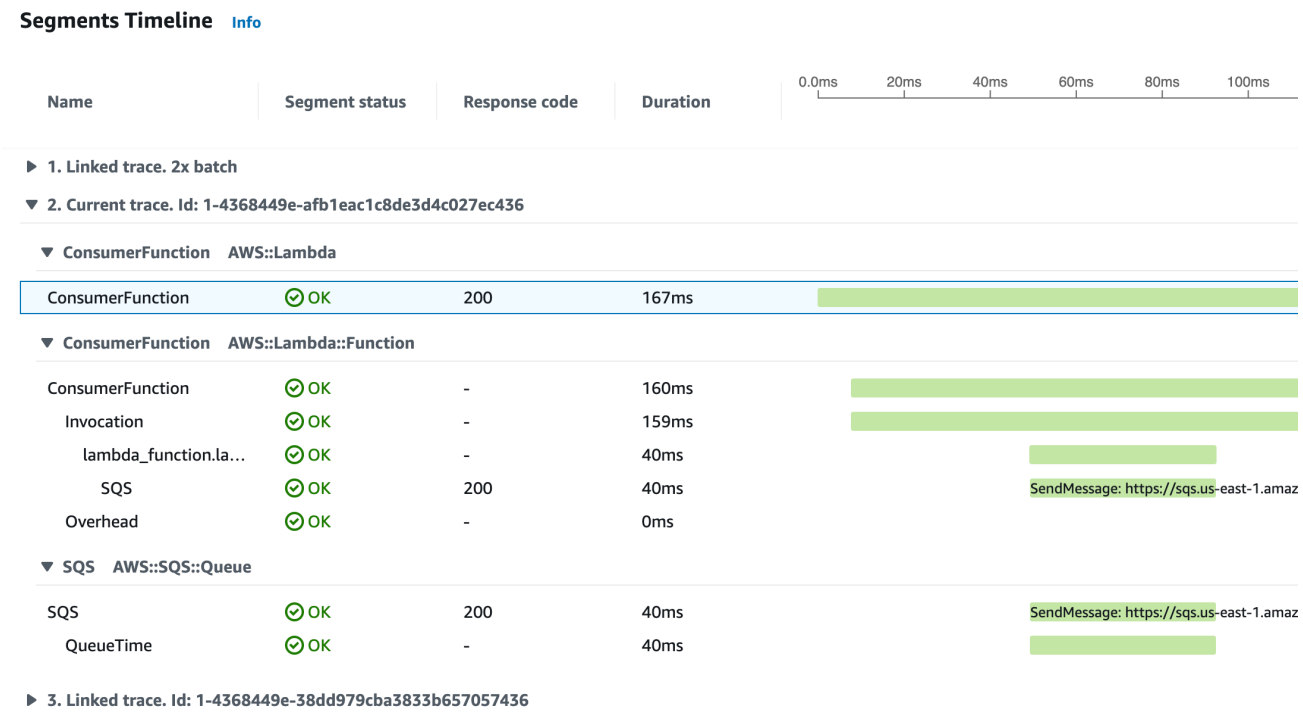
1. Usa Trace Map per selezionare un produttore di messaggi, Amazon SQS o un nodo consumer Lambda.
2. Scegli Visualizza tracce dal riquadro dei dettagli del nodo per visualizzare un elenco di tracce. Puoi anche accedere direttamente alla pagina Tracce all'interno della CloudWatch console.
3. Scegli una traccia specifica dall'elenco per aprire la pagina dei dettagli della traccia. La pagina dei dettagli della traccia visualizza un messaggio quando la traccia selezionata fa parte di un set di tracce collegato.



La mappa dei dettagli della traccia mostra la traccia corrente, insieme alle tracce collegate a monte e a valle, ognuna delle quali è contenuta in un riquadro che indica i limiti di ciascuna traccia. Se la traccia attualmente selezionata è collegata a più tracce collegate a monte o a valle, i nodi all'interno delle tracce collegate a monte o a valle vengono sovrapposti e viene visualizzato un pulsante Seleziona traccia.



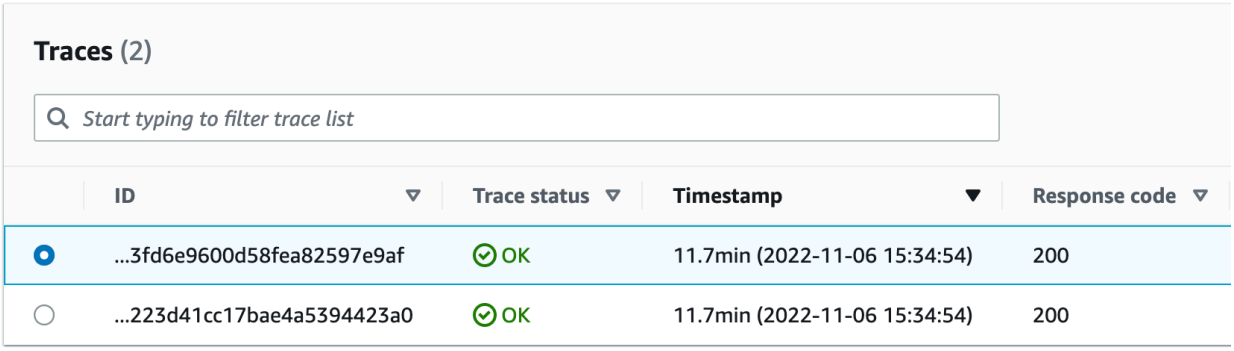
Sotto la mappa con i dettagli delle tracce, viene visualizzata una cronologia dei segmenti di traccia, comprese le tracce collegate a monte e a valle. Se sono presenti più tracce collegate a monte o a valle, i dettagli dei relativi segmenti non possono essere visualizzati. Per visualizzare i dettagli del segmento per una singola traccia all'interno di un insieme di tracce collegate, selezionate una singola traccia come descritto di seguito.



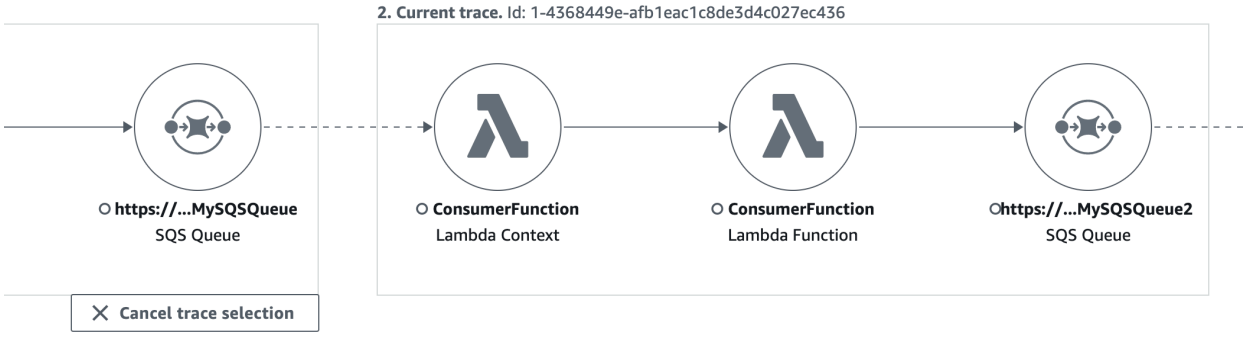
Seleziona una singola traccia all'interno di una serie di tracce collegate

Filtrate un insieme di tracce collegate a una singola traccia, per visualizzare i dettagli del segmento nella timeline.

1. Scegli Seleziona traccia sotto le tracce collegate sulla mappa dei dettagli della traccia. Viene visualizzato un elenco di tracce.



2. Seleziona il pulsante di opzione accanto a una traccia per visualizzarla all'interno della mappa dei dettagli della traccia.
3. Scegliete Annulla selezione traccia per visualizzare l'intero set di tracce collegate.



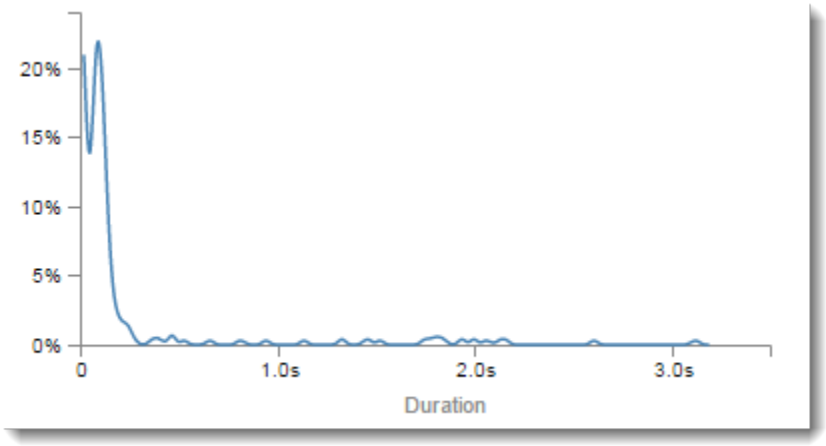
Utilizzo degli istogrammi di latenza

Quando si seleziona un nodo o un bordo su una mappa di AWS X-Ray traccia, la X-Ray console mostra un istogramma di distribuzione della latenza.

Latenza

La latenza è la quantità di tempo compresa tra il momento in cui una richiesta inizia e quello in cui viene completata. Un istogramma mostra una distribuzione di latenze. Mostra la durata sull'asse delle X e la percentuale di richieste che corrispondono a ciascuna durata sull'asse delle Y.

Questo istogramma mostra un servizio che completa la maggior parte delle richieste in meno di 300 ms. Una piccola percentuale di richieste richiedere fino a 2 secondi e alcuni valori erratici richiedono più tempo.



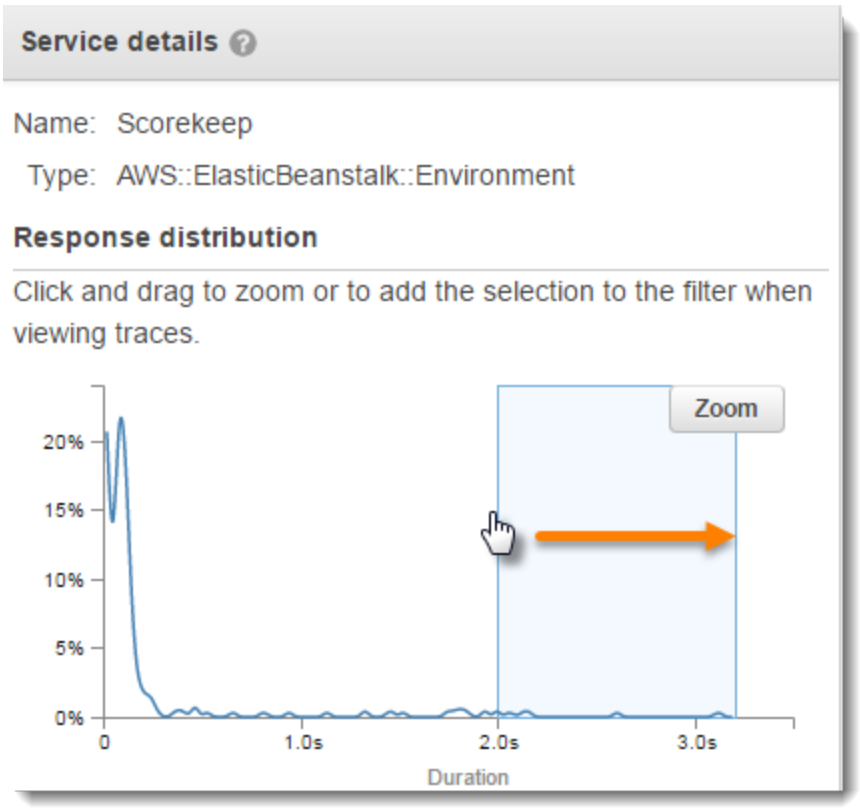
Interpretazione dei dettagli del servizio

Gli istogrammi del servizio e del collegamento forniscono una rappresentazione visiva della latenza dal punto di vista di un servizio o di un richiedente.

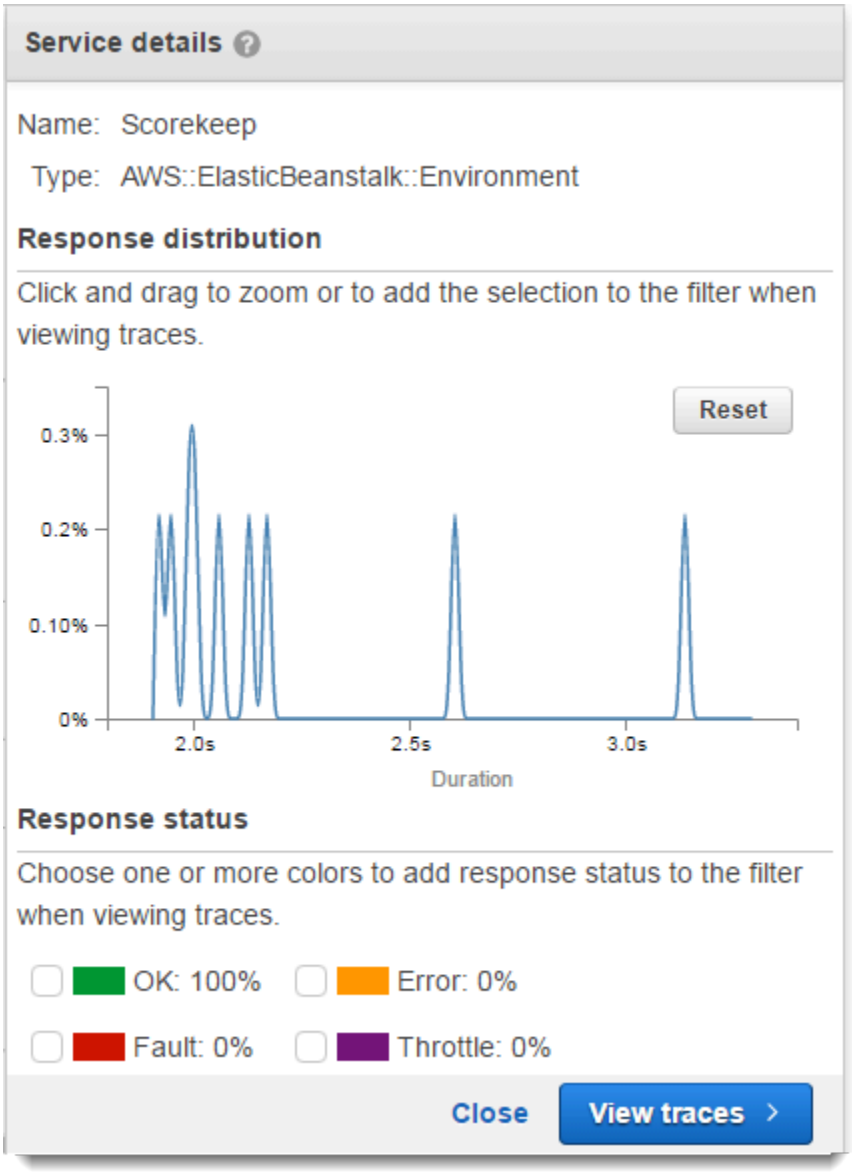
- Scegliete un nodo di servizio facendo clic sul cerchio. X-Ray mostra un istogramma per le richieste servite dal servizio. Le latenze sono quelle registrate dal servizio e non includono la latenza di rete tra il servizio e il richiedente.
- Scegli un collegamento facendo clic sulla linea o sulla punta della freccia del collegamento tra due servizi. X-Ray mostra un istogramma per le richieste dal richiedente che sono state elaborate dal servizio a valle. Le latenze sono quelle registrate dal richiedente e includono la latenza di rete tra i due servizi.

Per interpretare il pannello dell'istogramma dei Service details (Dettagli del servizio), puoi cercare i valori che differiscono maggiormente dalla gran parte dei valori nell'istogramma. Questi valori erratici possono essere visualizzati come picchi o punte nell'istogramma, ed è possibile visualizzare i tracciamenti relativi ad una specifica area per analizzare il problema.

Per visualizzare i tracciamenti filtrati per la latenza, seleziona un intervallo sull'istogramma. Fai clic sul punto in cui desideri avviare la selezione e trascina da sinistra verso destra per evidenziare un intervallo di latenze da includere nel filtro dei tracciamenti.



Dopo aver selezionato un intervallo, puoi scegliere Zoom per visualizzare solo quella parte di istogramma e restringere la selezione.



Una volta impostata l'area di interesse, scegli View traces (Visualizza tracciamenti).

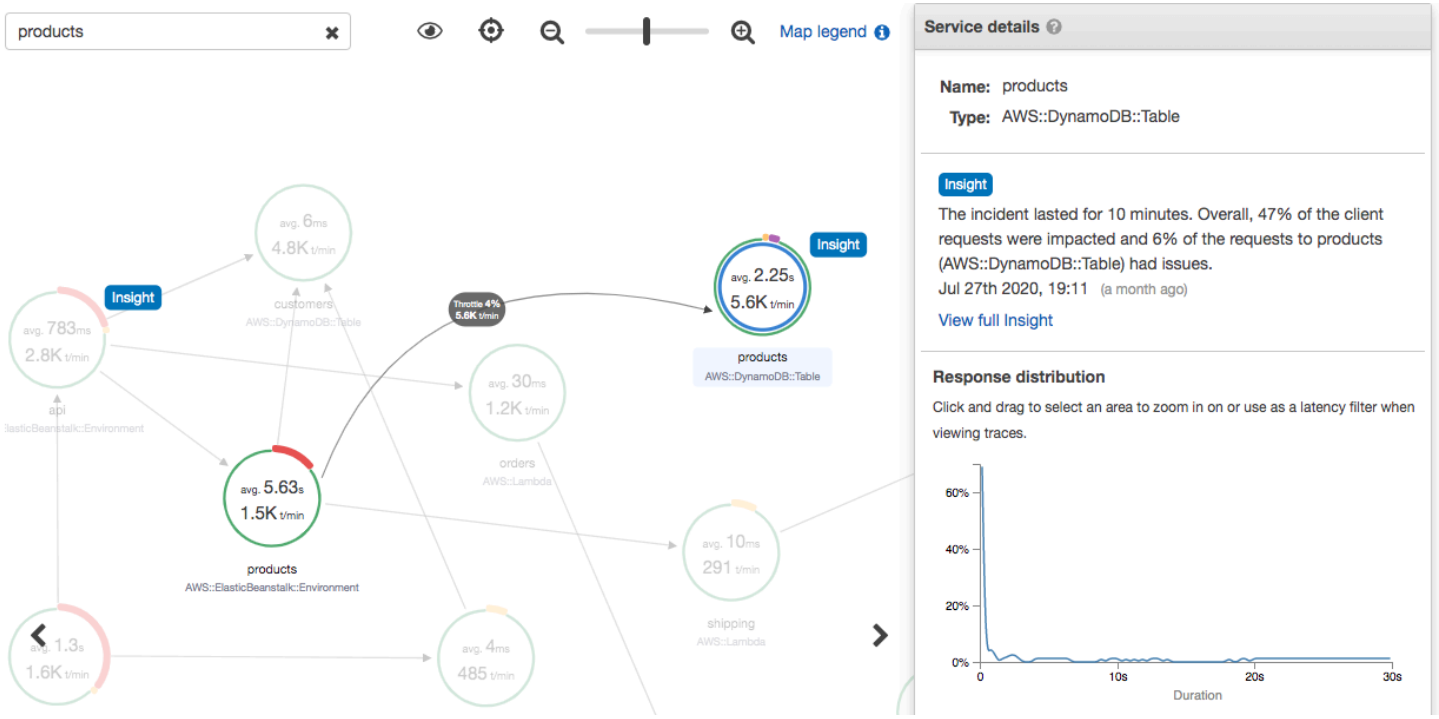
Utilizzo degli X-Ray approfondimenti

AWS X-Ray analizza continuamente i dati di tracciamento presenti nel tuo account per identificare i problemi emergenti nelle tue applicazioni. Quando i tassi di errore superano l'intervallo previsto, crea una panoramica che registra il problema e ne traccia l'impatto fino alla sua risoluzione. Con Insights, puoi:

- Identifica il punto in cui si verificano i problemi nell'applicazione, la causa principale del problema e l'impatto associato. L'analisi dell'impatto fornita da Insights consente di determinare la gravità e la priorità di un problema.

- Ricevi notifiche man mano che il problema cambia nel tempo. Le notifiche Insights possono essere integrate con la tua soluzione di monitoraggio e avviso tramite Amazon EventBridge. Questa integrazione consente di inviare e-mail o avvisi automatici in base alla gravità del problema.

La X-Ray console identifica i nodi con incidenti in corso nella mappa di tracciamento. Per visualizzare un riepilogo delle informazioni, scegli il nodo interessato. Puoi anche visualizzare e filtrare gli approfondimenti scegliendo Insights dal riquadro di navigazione a sinistra.



X-Ray crea un insight quando rileva un'anomalia in uno o più nodi della mappa dei servizi. Il servizio utilizza modelli statistici per prevedere i tassi di errore previsti dei servizi nell'applicazione. Nell'esempio precedente, l'anomalia è un aumento dei guasti da. AWS Elastic Beanstalk Il server Elastic Beanstalk ha subito diversi timeout delle chiamate API, causando un'anomalia nei nodi a valle.

Abilita X-Ray gli approfondimenti nella console

Gli approfondimenti devono essere abilitati per ogni gruppo con cui desideri utilizzare le funzionalità di Insights. Puoi abilitare gli approfondimenti dalla pagina Gruppi.

1. Apri la X-Ray console.

2. Seleziona un gruppo esistente o creane uno nuovo scegliendo Crea gruppo, quindi seleziona Abilita approfondimenti. Per ulteriori informazioni sulla configurazione dei gruppi nella X-Ray console, vedereConfigurazione dei gruppi.
3. Nel riquadro di navigazione a sinistra, scegli Insights, quindi scegli un approfondimento da visualizzare.



> **Note**
>
> X-Ray utilizza GetInsightSummaries, GetInsight GetInsightEvents, e operazioni GetInsightImpactGraph API per recuperare i dati dagli approfondimenti.
> Per ulteriori informazioni, consulta Come AWS X-Ray funziona con IAM.

Abilita le notifiche di approfondimenti

Con le notifiche di approfondimenti, viene creata una notifica per ogni evento di approfondimento, ad esempio quando un insight viene creato, cambia in modo significativo o viene chiuso. I clienti possono ricevere queste notifiche tramite Amazon EventBridge Events e utilizzare regole condizionali per intraprendere azioni come la notifica SNS, l'invocazione Lambda, la pubblicazione di messaggi su una coda SQS o su uno qualsiasi dei target supportati. EventBridge Le notifiche Insights vengono emesse nel migliore dei modi, ma non sono garantite. Per ulteriori informazioni sugli obiettivi, consulta Amazon EventBridge Targets.

Puoi abilitare le notifiche di Insights per qualsiasi gruppo abilitato a Insights dalla pagina Gruppi.

Per abilitare le notifiche per un X-Ray gruppo

1. Apri la X-Ray console.
2. Seleziona un gruppo esistente o creane uno nuovo scegliendo Crea gruppo, assicurati che Abilita approfondimenti sia selezionato, quindi seleziona Abilita notifiche. Per ulteriori informazioni sulla configurazione dei gruppi nella X-Ray console, consultaConfigurazione dei gruppi.

Per configurare le regole EventBridge condizionali di Amazon

1. Apri la EventBridge console https://console.aws.amazon.com/events/home Amazon.
2. Vai a Regole nella barra di navigazione a sinistra e scegli Crea regola.
3. Fornisci un nome e una descrizione per la regola.
4. Scegli Schema evento, quindi scegli Modello personalizzato. Fornisci uno schema contenente `"source": [ "aws.xray" ]` e`"detail-type": [ "AWS X-Ray Insight Update" ]`. Di seguito sono riportati alcuni esempi di modelli possibili.
   - Schema di eventi che corrisponda a tutti gli eventi in arrivo provenienti da X-Ray Insights:

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

   - Modello di evento che corrisponda a uno specifico **`state`** e**`category`**:

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

5. Seleziona e configura i target che desideri richiamare quando un evento soddisfa questa regola.
6. (Facoltativo) Fornisci tag per identificare e selezionare più facilmente questa regola.
7. Scegli Crea.

> **Note**
>
> X-Ray Le notifiche di insights inviano eventi ad Amazon EventBridge, che attualmente non supporta le chiavi gestite dal cliente. Per ulteriori informazioni, consulta Protezione dei dati in AWS X-Ray.

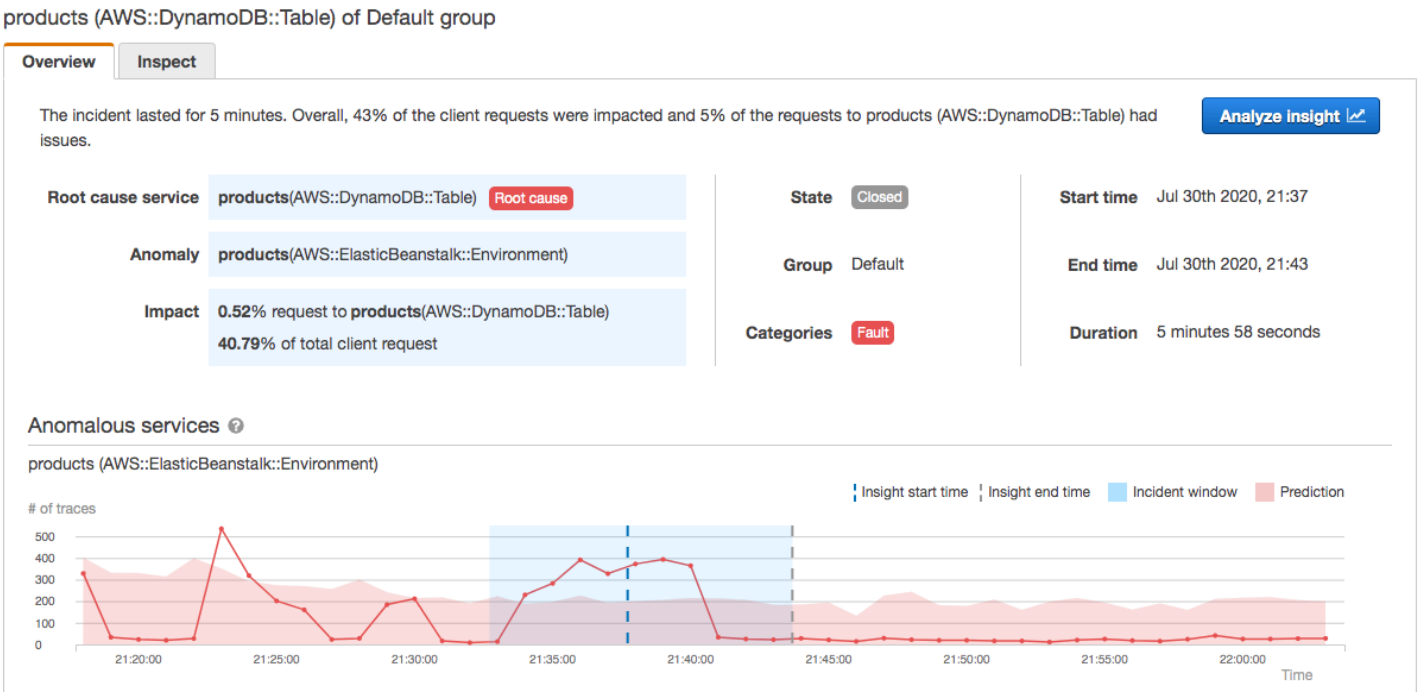
Panoramica di Insight

La pagina di panoramica per un approfondimento tenta di rispondere a tre domande chiave:

- Qual è il problema di fondo?
- Qual è la causa principale?
- Qual è l'impatto?

La sezione Servizi anomali mostra una cronologia per ciascun servizio che illustra la variazione dei tassi di guasto durante l'incidente. La timeline mostra il numero di tracce con guasti sovrapposti su una banda continua che indica il numero previsto di guasti in base alla quantità di traffico registrato. La durata dell'analisi è visualizzata dalla finestra Incidente. La finestra degli incidenti inizia quando X-Ray osserva che la metrica diventa anomala e persiste finché l'insight è attivo.

L'esempio seguente mostra un aumento dei guasti che hanno causato un incidente:



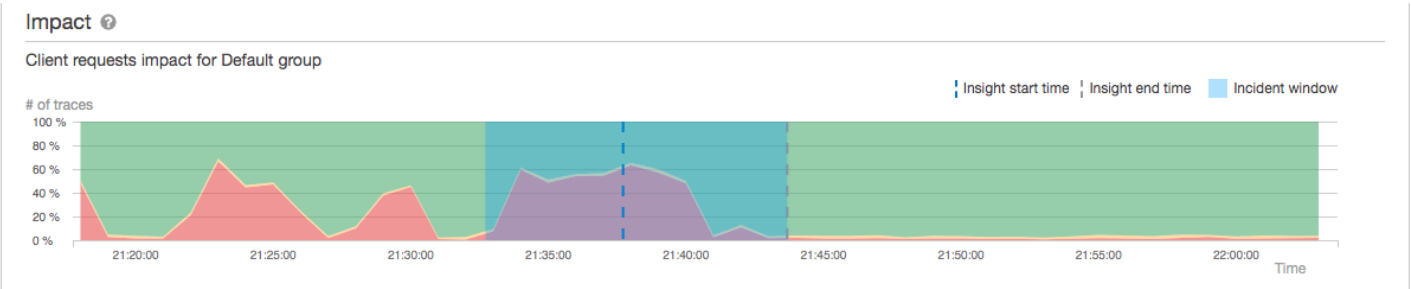
La sezione Root cause mostra una mappa di tracciamento incentrata sul servizio della causa principale e sul percorso interessato. È possibile nascondere i nodi non interessati selezionando l'icona a forma di occhio in alto a destra nella mappa della causa principale. Il root cause service è il nodo a valle più lontano in cui X-Ray è stata identificata un'anomalia. Può rappresentare un servizio che hai strumentato o un servizio esterno che il tuo servizio ha chiamato con un client strumentato.

Ad esempio, se chiami Amazon DynamoDB con un client AWS SDK strumentato, un aumento dei guasti di DynamoDB si traduce in una visione che vede DynamoDB come causa principale.

Per approfondire la causa principale, seleziona Visualizza i dettagli della causa principale nel grafico della causa principale. Puoi utilizzare la pagina Analytics per analizzare la causa principale e i messaggi correlati. Per ulteriori informazioni, consulta Interazione con la console di analisi.



I guasti che continuano a monte della mappa possono avere un impatto su più nodi e causare più anomalie. Se un errore viene trasmesso completamente all'utente che ha effettuato la richiesta, il risultato è un errore del client. Si tratta di un errore nel nodo principale della mappa di tracciamento. Il grafico Impact fornisce una cronologia dell'esperienza del cliente per l'intero gruppo. Questa esperienza viene calcolata in base alle percentuali dei seguenti stati: Fault, Error, Throttle e Okay.



Questo esempio mostra un aumento delle tracce con un guasto nel nodo principale durante il periodo dell'incidente. Gli incidenti nei servizi a valle non sempre corrispondono a un aumento degli errori dei clienti.

Scegliendo Analyze insight si apre la console di X-Ray Analytics in una finestra in cui è possibile approfondire l'insieme di tracce che causano l'insight. Per ulteriori informazioni, consulta Interazione con la console di analisi.

Comprendere l'impatto

AWS X-Ray misura l'impatto causato da un problema in corso nell'ambito della generazione di informazioni e notifiche. L'impatto è misurato in due modi:

- Impatto sul X-Ray gruppo
- Impatto sul servizio root cause

Questo impatto è determinato dalla percentuale di richieste che hanno esito negativo o che causano un errore entro un determinato periodo di tempo. Questa analisi dell'impatto consente di determinare la gravità e la priorità del problema in base allo scenario specifico. Questo impatto è disponibile come parte dell'esperienza della console oltre alle notifiche di Insights.

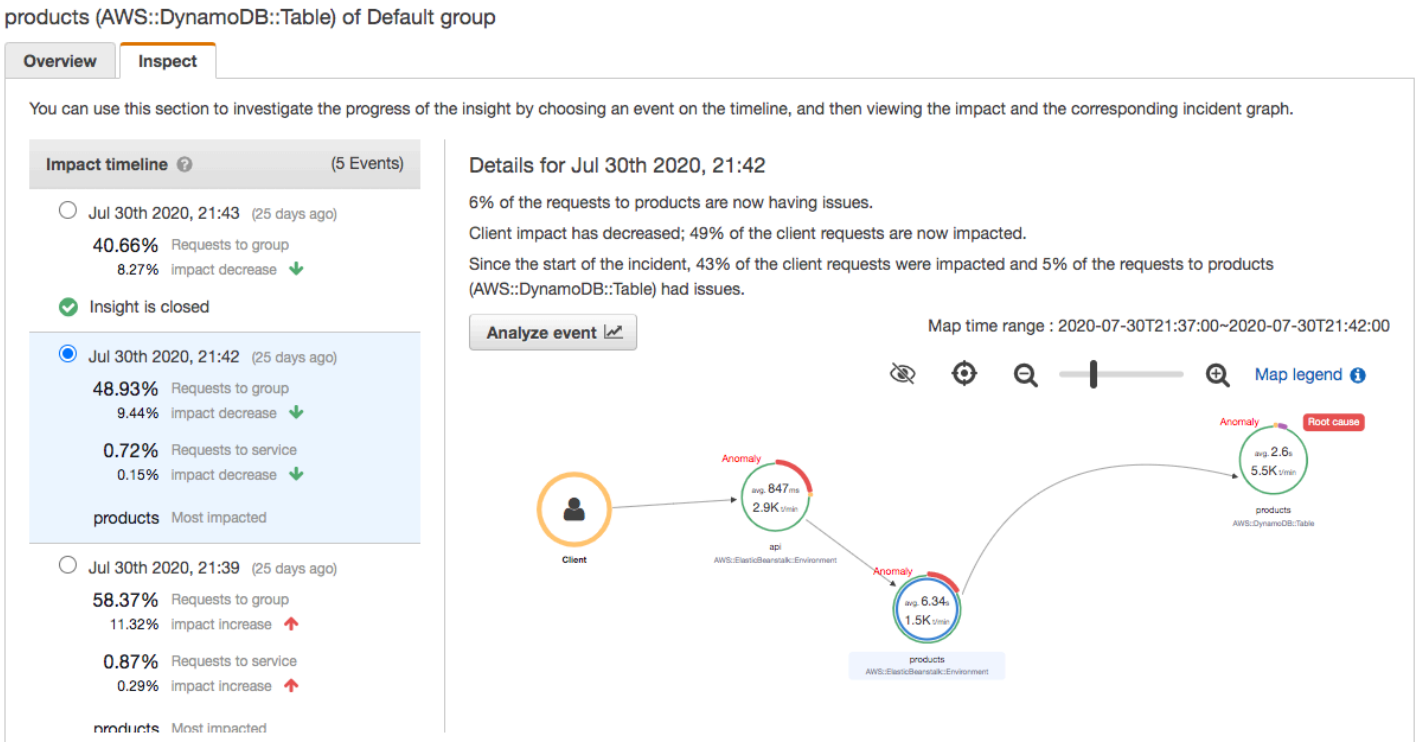
Deduplicazione

AWS X-Ray insights deduplica i problemi su più microservizi. Utilizza il rilevamento delle anomalie per determinare il servizio che è la causa principale di un problema, determina se altri servizi correlati presentano un comportamento anomalo dovuto alla stessa causa principale e registra il risultato come un'unica analisi.

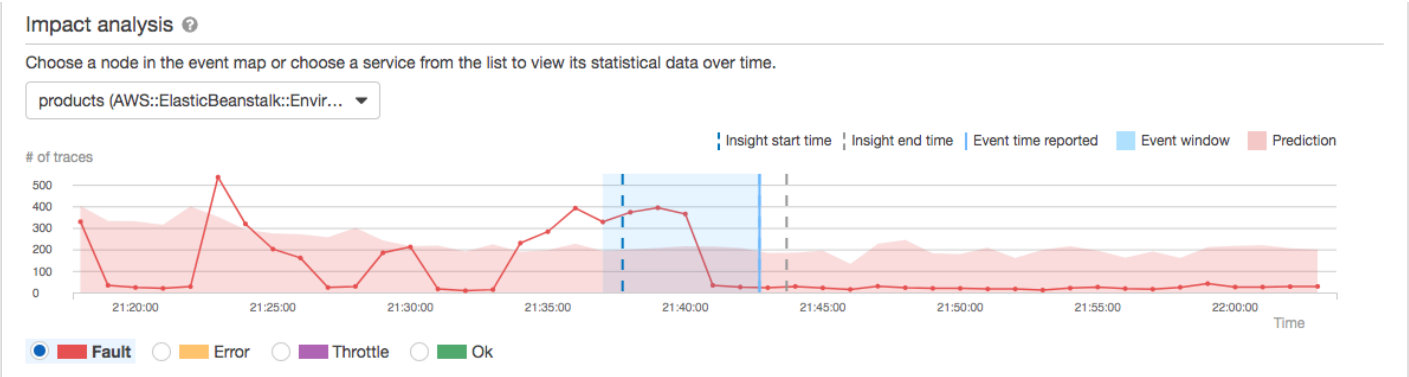
Esamina lo stato di avanzamento di un insight

X-Ray rivaluta periodicamente le informazioni fino a quando non vengono risolte e registra ogni modifica intermedia significativa come notifica, che può essere inviata come evento Amazon. EventBridge Ciò consente di creare processi e flussi di lavoro per determinare in che modo il problema è cambiato nel tempo e intraprendere le azioni appropriate, ad esempio l'invio di un'e-mail o l'integrazione con un sistema di avvisi utilizzando. EventBridge

Puoi esaminare gli eventi degli incidenti nella Timeline di impatto nella pagina Inspect. Per impostazione predefinita, la timeline mostra il servizio più interessato fino a quando non scegli un servizio diverso.



Per visualizzare una mappa di tracciamento e i grafici di un evento, sceglilo dalla timeline dell'impatto. La mappa di tracciamento mostra i servizi dell'applicazione interessati dall'incidente. In Impact analysis, i grafici mostrano le tempistiche dei guasti per il nodo selezionato e per i client del gruppo.



Per dare un'occhiata più approfondita alle tracce coinvolte in un incidente, scegli Analizza evento nella pagina Ispeziona. Puoi utilizzare la pagina Analytics per perfezionare l'elenco delle tracce e identificare gli utenti interessati. Per ulteriori informazioni, consulta Interazione con la console di analisi.

Interazione con la console di analisi

La console AWS X-Ray Analytics è uno strumento interattivo per l'interpretazione dei dati di tracciamento per comprendere rapidamente le prestazioni dell'applicazione e dei servizi sottostanti.

La console consente di esplorare, analizzare e visualizzare i tracciamenti tramite tempo di risposta interattivo e grafici di serie temporali.

Quando si effettuano selezioni nella console di analisi, la console crea dei filtri per riflettere il sottoinsieme selezionato di tutti i tracciamenti. È possibile definire il set di dati attivo con filtri sempre più granulari facendo clic sui grafici e i pannelli di parametri e campi associati con il set di tracciamenti corrente.

Argomenti

- Caratteristiche della console
- Distribuzione del tempo di risposta
- Attività delle serie temporali
- Esempi di flussi di lavoro
- Osservare gli errori sul grafico di servizio
- Identificare i picchi del tempo di risposta
- Visualizzare tutti le tracce contrassegnate con un codice di stato
- Visualizzare tutti gli elementi in un sottogruppo e associati a un utente
- Confrontare due set di tracce con criteri diversi
- Identificare un tracciamento rilevante e visualizzare i dettagli

Caratteristiche della console

La console X-Ray Analytics utilizza le seguenti funzionalità chiave per raggruppare, filtrare, confrontare e quantificare i dati di traccia.

Funzionalità

Funzionalità	Description
Groups (Gruppi)	Il gruppo selezionato inizialmente è `Default`. Per modificare il gruppo recuperato, seleziona re un gruppo diverso dal menu a destra della barra di ricerca dell'espressione di filtro principale. Per ulteriori informazioni sui gruppi, consulta Utilizzo delle espressioni filtro con i gruppi.

Funzionalità	Description
Retrieved traces (Tracciamenti recuperati)	Per impostazione predefinita, la console di analisi genera grafici in base a tutti i tracciamenti nel gruppo selezionato. I tracciamenti recuperati rappresentano tutti i tracciamenti nel set di lavoro. Puoi trovare il conteggio dei tracciamenti in questo riquadro. Le espressioni di filtro applicate alla barra di ricerca principale definiscono e aggiornano i tracciamenti recuperati.
Mostra dai grafici charts/Hide	Un elemento di attivazione/disattivazione per confrontare il gruppo attivo rispetto ai tracciamenti recuperati. Per confrontare i dati relativi al gruppo rispetto a tutti i filtri attivi, scegliere Show in charts (Mostra in grafici). Per rimuovere questa visualizzazione dai grafici, scegliere Hide from charts (Nascondi da grafici).
Filtered trace set A (Set di tracciamenti filtrato A)	Attraverso le interazioni con i grafici e le tabelle, applica i filtri per creare i criteri per il set di tracce filtrato A. Man mano che i filtri vengono applicati, il numero di tracce applicabili e la percentuale di tracce dal totale recuperate vengono calcolati all'interno di questo riquadro. I filtri sono inseriti come tag all'interno del riquadro Filtered trace set A (Set di tracce filtrate A e possono anche essere rimossi dal riquadro.

Funzionalità	Description
Refine (Definisci)	Questa funzione aggiorna il set di tracce recuperate in base ai filtri applicati al set di tracce A. Il perfezionamento del set di tracce recuperate aggiorna il working set di tutte le tracce recuperate in base ai filtri per il set di tracce A. Il working set di tracce recuperate è un sottoinsieme campionato di tutte le tracce nel gruppo.
Filtered trace set B (Set di tracciamenti filtrato B)	Una volta creato, il set di tracce filtrato B è una copia del set di tracce filtrato A. Per confrontare i due set di tracce, effettuate nuove selezioni di filtro che verranno applicate al set di tracce B, mentre il set di tracce A rimane fisso. Quando i filtri vengono applicati , il numero di tracciamenti applicabili e la percentuale di tracciamenti dal totale recuperat i sono calcolati all'interno di questo riquadro. I filtri sono inseriti come tag all'interno del riquadro Filtered trace set B (Set di tracce filtrate B) e possono anche essere rimossi dal riquadro.
Response Time Root Cause Entity Paths (Percorsi di entità causa principale del tempo di risposta)	Una tabella dei percorsi delle entità registrate. X-Ray determina quale percorso nella traccia è la causa più probabile del tempo di risposta. Il formato indica una gerarchia di entità incontrate che termina in una causa principale del tempo di risposta. Utilizza queste righe per filtrare gli errori ricorrenti del tempo di risposta. Per ulteriori informazioni sulla personalizzazione di un filtro di causa principale e su come ottenere i dati tramite l'API, consultare l'argomento relativo al recupero e definizione dell’analisi delle cause principali.

Funzionalità	Description
Delta (Δ)	Una colonna che viene aggiunta alle tabelle di parametri quando sia il set di tracce A che il set di tracce B sono attivi. La colonna del Delta calcola la differenza nella percentuale di tracce tra il set di tracce A e il set di tracce B.

Distribuzione del tempo di risposta

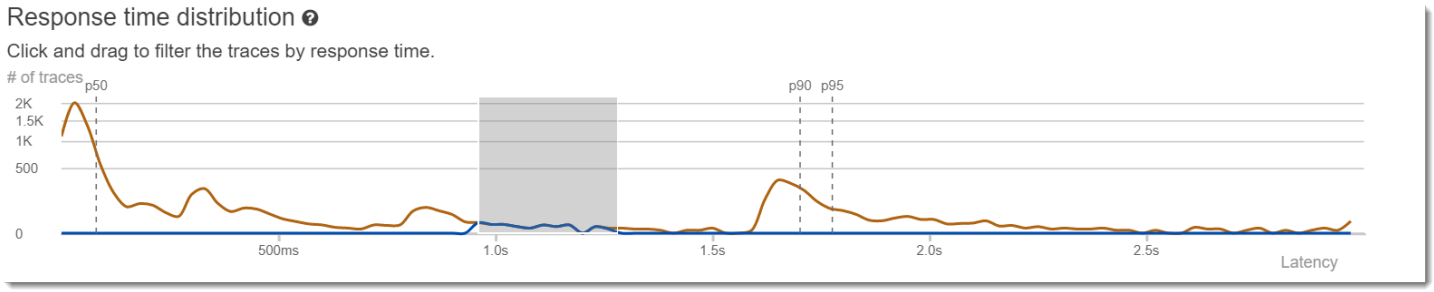
La console X-Ray Analytics genera due grafici principali per aiutarti a visualizzare le tracce: distribuzione dei tempi di risposta e attività delle serie temporali. In questa sezione e nella seguente sono forniti esempi di ognuno e le informazioni di base per leggere i grafici.

Di seguito sono elencati i colori associati al grafico della linea temporale di risposta (il grafico delle serie temporali impiega la stessa combinazione di colori):

- Tutte le tracce del gruppo: grigie
- Tracce recuperate — arancione
- Set di tracce filtrate A — verde
- Set di tracce filtrate B — blu

Example— Distribuzione dei tempi di risposta

La distribuzione del tempo di risposta è un grafico che mostra il numero di tracciamenti con un determinato tempo di risposta. Fare clic e trascinare per effettuare selezioni all'interno della distribuzione del tempo di risposta. Questa operazione seleziona e crea un filtro sul set di tracciamenti di lavoro denominato `responseTime` per tutti i tracciamenti all'interno di un determinato tempo di risposta.

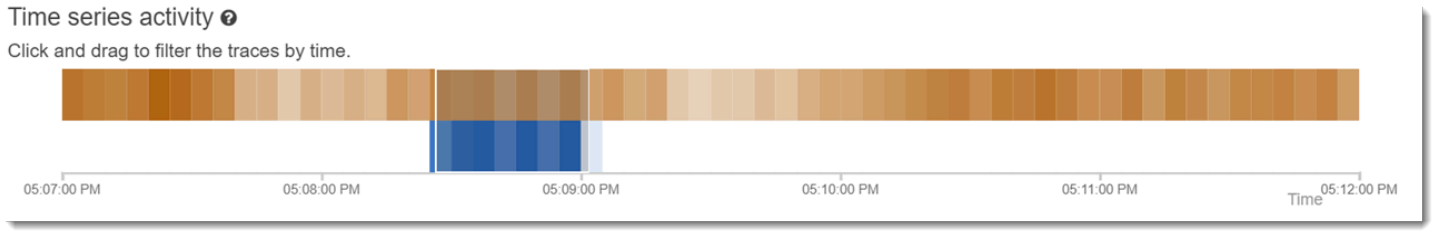


Attività delle serie temporali

Il grafico dell'attività di serie temporali mostra il numero di tracciamenti in un determinato intervallo di tempo. Gli indicatori di colore rispecchiano i colore del grafico a linee della distribuzione del tempo di risposta. Più è scuro e pieno il blocco di colore all'interno della serie dell'attività, più sono i tracciamenti rappresentati in quel determinato momento.

Example— Attività delle serie temporali

Fare clic e trascinare per effettuare selezioni all'interno del grafico di attività delle serie temporali. Questa operazione seleziona e crea un filtro denominato `timerange` sul set di tracciamenti di lavoro per tutti i tracciamenti all'interno di uno specifico intervallo di tempo.



Esempi di flussi di lavoro

Gli esempi seguenti mostrano i casi d'uso più comuni della console X-Ray Analytics. Ogni esempio illustra una funzione chiave dell'esperienza della console. Come gruppo, gli esempi seguono un flusso di lavoro di risoluzione dei problemi di base. La procedura illustra come individuare i nodi non integri e quindi come interagire con la console di Analytics per generare automaticamente query comparative. Una volta ristretto l'ambito attraverso le query, sarà possibile visualizzare finalmente i dettagli delle tracce di interesse per determinare cosa sta danneggiando lo stato del servizio.

Osservare gli errori sul grafico di servizio

La mappa di tracciamento indica lo stato di salute di ciascun nodo colorandolo in base al rapporto tra chiamate riuscite, errori e guasti. Quando viene visualizzata una percentuale di rosso sul nodo, è segnalato un errore. Utilizzare la console di analisi X-Ray per indagare sull'errore.

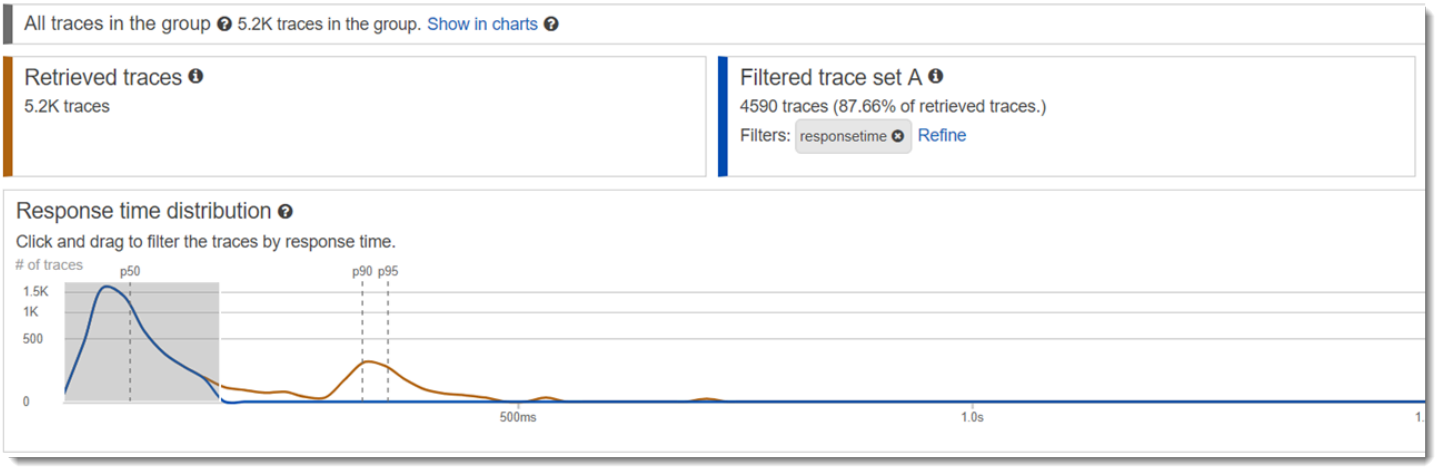
Per ulteriori informazioni su come leggere la mappa di traccia, vedere Visualizzazione della mappa di traccia.



Identificare i picchi del tempo di risposta

Utilizzando la distribuzione del tempo di risposta, è possibile osservare i picchi nel tempo di risposta. Selezionando questo picco nel tempo di risposta, le tabelle sotto i grafici si aggiorneranno per esporre tutti i parametri associati, come i codici di stato.

Quando fai clic e trascini, X-Ray seleziona e crea un filtro. Viene mostrato in un'ombreggiatura grigia sulla parte superiore delle righe del grafico. È ora possibile trascinare l'ombreggiatura a sinistra e destra lungo la distribuzione per aggiornare la selezione e il filtro.



Visualizzare tutti le tracce contrassegnate con un codice di stato

È possibile esaminare i dettagli dei tracciamenti all'interno del picco selezionato utilizzando le tabelle di parametri sotto ai grafici. Facendo clic su una riga nella tabella HTTP STATUS CODE, viene creato automaticamente un filtro sul set di dati di lavoro. Ad esempio, è possibile visualizzare tutti i tracciamenti del codice di stato 500. In questo modo viene creato un tag di filtro nel riquadro del set di tracciamenti denominato `http.status`.

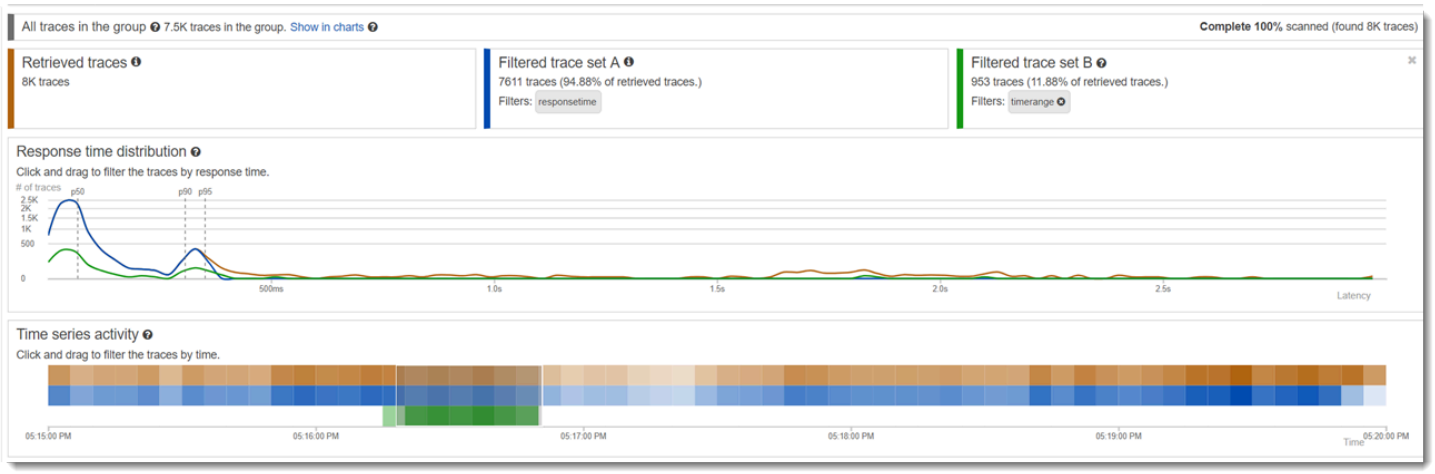
Visualizzare tutti gli elementi in un sottogruppo e associati a un utente

Analizzare il set di errori in base a utente, URL, causa principale del tempo di risposta o altri attributi predefiniti. Ad esempio, per filtrare ulteriormente il set di tracciamenti con un codice di stato 500, selezionare una riga dalla tabella USERS. Questo crea due tag di filtro nel riquadro del set di tracciamenti: `http.status`, come indicato in precedenza, e `user`.

Confrontare due set di tracce con criteri diversi

Confrontare i diversi utenti e le relative richieste POST per trovare altre discrepanze e correlazioni. Applicare il primo set di filtri. Questi sono definiti da una linea blu nella distribuzione del tempo di risposta. Quindi selezionare Compare (Confronta). Inizialmente, questo crea una copia dei filtri nel set di tracciamenti A.

Per procedere, definire un nuovo set di filtri da applicare al set di tracciamenti B. Questo secondo set è rappresentato da una linea verde. L'esempio seguente mostra le diverse linee in base allo schema di colori blu e verde.



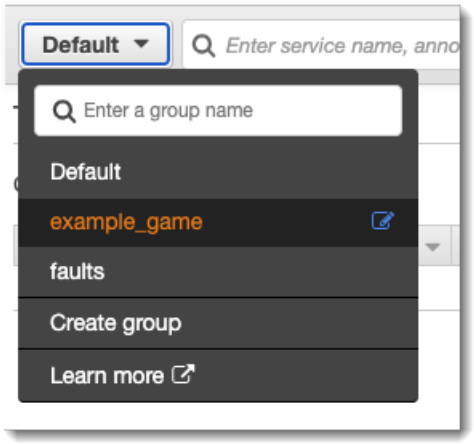
Identificare un tracciamento rilevante e visualizzare i dettagli

Quando si delimita l'ambito usando i filtri della console, l'elenco dei tracciamenti sotto le tabelle di parametri diventa più significativo. La tabella dell'elenco delle tracce unisce le informazioni su URL, USER e STATUS CODE in un'unica visualizzazione. Per informazioni più approfondite, selezionare una riga di questa tabella per aprire la pagina dei dettagli del tracciamento e visualizzare la sequenza temporale e i dati non elaborati.

Configurazione dei gruppi

I gruppi costituiscono una raccolta di tracciamenti definiti da un'espressione di filtro. Puoi utilizzare i gruppi per generare grafici di servizio aggiuntivi e fornire CloudWatch metriche Amazon. Puoi utilizzare la AWS X-Ray console o l'API X-Ray per creare e gestire gruppi per i tuoi servizi. Questo argomento descrive come creare e gestire gruppi utilizzando la console X-Ray. Per informazioni su come gestire i gruppi utilizzando l'API X-Ray, vedere. Gruppi

È possibile creare gruppi di tracce per mappe di traccia, tracce o analisi. Quando crei un gruppo, il gruppo diventa disponibile come filtro nel menu a discesa del gruppo su tutte e tre le pagine: Trace Map, Traces e Analytics.



I gruppi sono identificati dal nome o da un Amazon Resource Name (ARN) e contengono un'espressione di filtro. Il servizio confronta i tracciamenti in entrata con l'espressione e le memorizza di conseguenza. Per ulteriori informazioni su come creare un'espressione di filtro, vedereUtilizzo delle espressioni di filtro.

L'aggiornamento dell'espressione di filtro di un gruppo non modifica i dati già registrati. L'aggiornamento si applica solo per le successive tracce. Ciò può comportare un grafo unito delle espressioni nuove e vecchie. Per evitare ciò, eliminate un gruppo corrente e createne uno nuovo.

> **Note**
>
> I gruppi sono fatturati per il numero di tracce recuperate corrispondenti all'espressione di filtro. Per ulteriori informazioni, consulta Prezzi di AWS X-Ray.

Argomenti

- Creazione di un gruppo
- Applica un gruppo
- Modifica un gruppo
- Clonare un gruppo
- Eliminazione di un gruppo
- Visualizza le metriche di gruppo in Amazon CloudWatch

Creazione di un gruppo

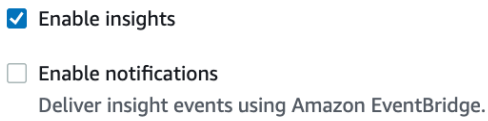
> **Note**
>
> Ora puoi configurare i gruppi X-Ray direttamente dalla console Amazon CloudWatch . È inoltre possibile continuare a utilizzare la console X-Ray.

CloudWatch console

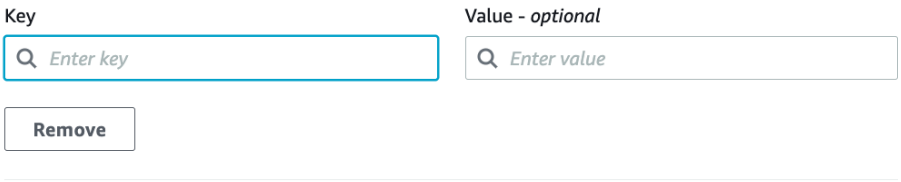
1. Accedi a Console di gestione AWS e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli Impostazioni nel riquadro di navigazione a sinistra.
3. Scegli Visualizza impostazioni in Gruppi all'interno della sezione Tracce X-Ray.
4. Scegli Crea gruppo sopra l'elenco dei gruppi.
5. Nella pagina Crea gruppo, inserisci un nome per il gruppo. Il nome di un gruppo può avere un massimo di 32 caratteri e contenere caratteri alfanumerici e trattini. I nomi dei gruppi distinguono tra maiuscole e minuscole
6. Immettere un'espressione di filtro. Per ulteriori informazioni su come creare un'espressione di filtro, vedereUtilizzo delle espressioni di filtro. Nell'esempio seguente, il gruppo filtra le tracce di errore provenienti dal servizio `api.example.com` e le richieste al servizio in cui il tempo di risposta è stato maggiore o uguale a cinque secondi.

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

7. In Insights, abilita o disabilita l'accesso a Insights per il gruppo. Per ulteriori informazioni sulle informazioni dettagliate, consulta Utilizzo degli X-Ray approfondimenti.

8. In Tag, scegli Aggiungi nuovo tag per inserire una chiave di tag e, facoltativamente, un valore per il tag. Continua ad aggiungere altri tag, se lo desideri. Le chiavi dei tag devono essere uniche. Per eliminare un tag, scegli Rimuovi sotto ogni tag. Per ulteriori informazioni sui tag, consulta Etichettatura delle regole e dei gruppi di campionamento a raggi X.



9. Seleziona Crea gruppo.

X-Ray console

1. Accedi Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Apri la pagina Crea gruppo dalla pagina Gruppi nel riquadro di navigazione a sinistra o dal menu di gruppo in una delle seguenti pagine: Trace Map, Traces e Analytics.
3. Nella pagina Crea gruppo, inserisci un nome per il gruppo. Il nome di un gruppo può avere un massimo di 32 caratteri e contenere caratteri alfanumerici e trattini. I nomi dei gruppi distinguono tra maiuscole e minuscole
4. Immettere un'espressione di filtro. Per ulteriori informazioni su come creare un'espressione di filtro, vedereUtilizzo delle espressioni di filtro. Nell'esempio seguente, il gruppo filtra le tracce di errore provenienti dal servizio `api.example.com` e le richieste al servizio in cui il tempo di risposta è stato maggiore o uguale a cinque secondi.

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

5. In Insights, abilita o disabilita l'accesso a Insights per il gruppo. Per ulteriori informazioni sulle informazioni dettagliate, consulta Utilizzo degli X-Ray approfondimenti.

6. In Tag, inserisci una chiave di tag e, facoltativamente, un valore del tag. Quando aggiungi un tag, viene visualizzata una nuova riga in cui inserire un altro tag. Le chiavi dei tag devono essere uniche. Per eliminare un tag, scegli X alla fine della riga del tag. Per ulteriori informazioni sui tag, consulta Etichettatura delle regole e dei gruppi di campionamento a raggi X.



7. Seleziona Crea gruppo.

Applica un gruppo

CloudWatch console

1. Accedi a Console di gestione AWS e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Apri una delle seguenti pagine dal riquadro di navigazione sotto Tracce X-Ray:
   - Mappa di tracciamento
   - Tracce
3. Immettete il nome di un gruppo nel filtro di gruppo Filtra per X-Ray. I dati mostrati nella pagina vengono modificati in base all'espressione di filtro impostata nel gruppo.

X-Ray console

1. Accedi Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Apri una delle seguenti pagine dal pannello di navigazione:
   - Mappa di tracciamento
   - Tracce
   - Analisi
3. Nel menu del gruppo, scegli il gruppo in cui hai creatothe section called “Creazione di un gruppo”. I dati mostrati nella pagina vengono modificati in base all'espressione di filtro impostata nel gruppo.

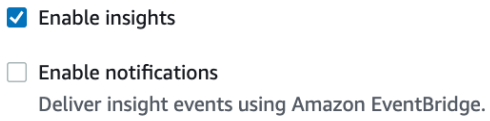
Modifica un gruppo

CloudWatch console

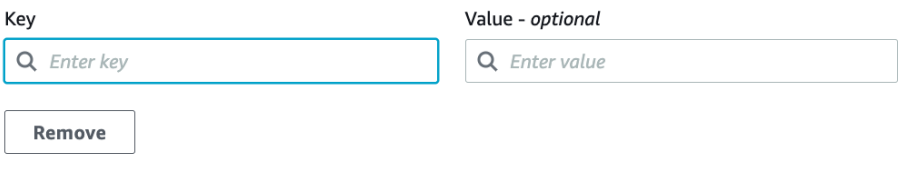
1. Accedi a Console di gestione AWS e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli Impostazioni nel riquadro di navigazione a sinistra.
3. Scegli Visualizza impostazioni in Gruppi all'interno della sezione Tracce X-Ray.
4. Scegliete un gruppo dalla sezione Gruppi, quindi scegliete Modifica.
5. Sebbene non sia possibile rinominare un gruppo, è possibile aggiornare l'espressione del filtro. Per ulteriori informazioni su come creare un'espressione di filtro, consultaUtilizzo delle espressioni di filtro. Nell'esempio seguente, il gruppo filtra le tracce di errore provenienti dal servizio`api.example.com`, in cui è contenuto l'indirizzo URL della richiesta`example/game`, e il tempo di risposta per le richieste è stato maggiore o uguale a cinque secondi.

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

6. In Insights, abilita o disabilita l'accesso a Insights per il gruppo. Per ulteriori informazioni sulle informazioni dettagliate, consulta Utilizzo degli X-Ray approfondimenti.

7. In Tag, scegli Aggiungi nuovo tag per inserire una chiave di tag e, facoltativamente, un valore per il tag. Continua ad aggiungere altri tag, se lo desideri. Le chiavi dei tag devono essere uniche. Per eliminare un tag, scegli Rimuovi sotto ogni tag. Per ulteriori informazioni sui tag, consulta Etichettatura delle regole e dei gruppi di campionamento a raggi X.

8. Quando hai finito di aggiornare il gruppo, scegli Aggiorna gruppo.

X-Ray console

1. Accedi Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Effettua una delle seguenti operazioni per aprire la pagina Modifica gruppo.

   a. Nella pagina Gruppi, scegli il nome di un gruppo per modificarlo.

   b. Nel menu del gruppo in una delle pagine seguenti, posiziona il puntatore su un gruppo, quindi scegli Modifica.

   - Traccia la mappa
   - Tracce
   - Analisi

3. Sebbene non sia possibile rinominare un gruppo, è possibile aggiornare l'espressione del filtro. Per ulteriori informazioni su come creare un'espressione di filtro, consultaUtilizzo delle espressioni di filtro. Nell'esempio seguente, il gruppo filtra le tracce di errore provenienti dal servizio`api.example.com`, in cui è contenuto l'indirizzo URL della richiesta`example/game`, e il tempo di risposta per le richieste è stato maggiore o uguale a cinque secondi.

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

4. In Insights, abilita o disabilita gli approfondimenti e le notifiche di approfondimenti per il gruppo. Per ulteriori informazioni sulle informazioni dettagliate, consulta Utilizzo degli X-Ray approfondimenti.



5. In Tag, modifica le chiavi e i valori dei tag. Le chiavi dei tag devono essere uniche. I valori dei tag sono facoltativi; puoi eliminarli, se lo desideri. Per eliminare un tag, scegli X alla fine della riga del tag. Per ulteriori informazioni sui tag, consulta Etichettatura delle regole e dei gruppi di campionamento a raggi X.



6. Quando hai finito di aggiornare il gruppo, scegli Aggiorna gruppo.

Clonare un gruppo

La clonazione di un gruppo crea un nuovo gruppo con l'espressione di filtro e i tag di un gruppo esistente. Quando clonate un gruppo, il nuovo gruppo ha lo stesso nome del gruppo da cui è stato clonato, con l'`-clone`aggiunta del nome.

CloudWatch console

1. Accedi a Console di gestione AWS e apri la console all'indirizzo. CloudWatch https://console.aws.amazon.com/cloudwatch/
2. Scegli Impostazioni nel riquadro di navigazione a sinistra.
3. Scegli Visualizza impostazioni in Gruppi all'interno della sezione Tracce X-Ray.
4. Scegli un gruppo dalla sezione Gruppi, quindi scegli Clona.
5. Nella pagina Crea gruppo, il nome del gruppo è *`group-name`*`-clone`. Facoltativamente, inserisci un nuovo nome per il gruppo. Il nome di un gruppo può avere un massimo di 32 caratteri e contenere caratteri alfanumerici e trattini. I nomi dei gruppi distinguono tra maiuscole e minuscole
6. È possibile mantenere l'espressione di filtro dal gruppo esistente o, facoltativamente, immettere una nuova espressione di filtro. Per ulteriori informazioni su come creare un'espressione di filtro, vedereUtilizzo delle espressioni di filtro. Nell'esempio seguente, il gruppo filtra le tracce di errore provenienti dal servizio `api.example.com` e le richieste al servizio in cui il tempo di risposta è stato maggiore o uguale a cinque secondi.

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

7. In Tag, modificate le chiavi e i valori dei tag, se necessario. Le chiavi dei tag devono essere uniche. I valori dei tag sono facoltativi; puoi eliminarli se lo desideri. Per eliminare un tag, scegli X alla fine della riga del tag. Per ulteriori informazioni sui tag, consulta Etichettatura delle regole e dei gruppi di campionamento a raggi X.
8. Seleziona Crea gruppo.

X-Ray console

1. Accedi Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Apri la pagina Gruppi dal riquadro di navigazione a sinistra e scegli il nome del gruppo che desideri clonare.
3. Scegli il gruppo Clone dal menu Azioni.
4. Nella pagina Crea gruppo, il nome del gruppo è *`group-name`*`-clone`. Facoltativamente, inserisci un nuovo nome per il gruppo. Il nome di un gruppo può avere un massimo di 32 caratteri e contenere caratteri alfanumerici e trattini. I nomi dei gruppi distinguono tra maiuscole e minuscole
5. È possibile mantenere l'espressione di filtro dal gruppo esistente o, facoltativamente, immettere una nuova espressione di filtro. Per ulteriori informazioni su come creare un'espressione di filtro, vedereUtilizzo delle espressioni di filtro. Nell'esempio seguente, il gruppo filtra le tracce di errore provenienti dal servizio `api.example.com` e le richieste al servizio in cui il tempo di risposta è stato maggiore o uguale a cinque secondi.

   ```
   service("api.example.com") { fault = true OR responsetime >= 5 }
   ```
6. In Tag, modificate le chiavi e i valori dei tag, se necessario. Le chiavi dei tag devono essere uniche. I valori dei tag sono facoltativi; puoi eliminarli se lo desideri. Per eliminare un tag, scegli X alla fine della riga del tag. Per ulteriori informazioni sui tag, consulta Etichettatura delle regole e dei gruppi di campionamento a raggi X.
7. Seleziona Crea gruppo.

Eliminazione di un gruppo

Segui i passaggi descritti in questa sezione per eliminare un gruppo. Non puoi eliminare il gruppo predefinito.

CloudWatch console

1. Accedi a Console di gestione AWS e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli Impostazioni nel riquadro di navigazione a sinistra.
3. Scegli Visualizza impostazioni in Gruppi all'interno della sezione Tracce X-Ray.

4. Scegliete un gruppo dalla sezione Gruppi, quindi scegliete Elimina.
5. Quando ti viene richiesto di confermare, scegli Elimina.

X-Ray console

1. Accedi Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Apri la pagina Gruppi dal riquadro di navigazione a sinistra e scegli il nome del gruppo che desideri eliminare.
3. Nel menu Azioni, scegli Elimina gruppo.
4. Quando ti viene richiesto di confermare, scegli Elimina.

Visualizza le metriche di gruppo in Amazon CloudWatch

Dopo la creazione di un gruppo, le tracce in entrata vengono confrontate con l'espressione del filtro del gruppo mentre vengono archiviate nel servizio X-Ray. Le metriche relative al numero di tracce che soddisfano ogni criterio vengono pubblicate su Amazon CloudWatch ogni minuto. Scegliendo Visualizza metrica nella pagina Modifica gruppo si apre la CloudWatch console alla pagina Metrica. Per ulteriori informazioni su come utilizzare i CloudWatch parametri, consulta Using Amazon CloudWatch Metrics nella Amazon CloudWatch User Guide.

CloudWatch console

1. Accedi Console di gestione AWS e apri la CloudWatch console all'indirizzo. https://console.aws.amazon.com/cloudwatch/
2. Scegli Impostazioni nel riquadro di navigazione a sinistra.
3. Scegli Visualizza impostazioni in Gruppi all'interno della sezione Tracce X-Ray.
4. Scegliete un gruppo dalla sezione Gruppi, quindi scegliete Modifica.
5. Nella pagina Modifica gruppo, scegli Visualizza metrica.

   La pagina Metriche della CloudWatch console si apre in una nuova scheda.

X-Ray console

1. Accedi Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.

2. Apri la pagina Gruppi dal riquadro di navigazione a sinistra, quindi scegli il nome del gruppo di cui desideri visualizzare le metriche.
3. Nella pagina Modifica gruppo, scegli Visualizza metrica.

   La pagina Metriche della CloudWatch console si apre in una nuova scheda.

Configurazione delle regole di campionamento di

Puoi utilizzare la AWS X-Ray console per configurare le regole di campionamento per i tuoi servizi. AWS Distro for OpenTelemetry, X-Ray SDK Servizi AWS e che supportano il tracciamento attivo con configurazioni di campionamento utilizzano regole di campionamento per determinare quali richieste registrare.

Argomenti

- Configurazione delle regole di campionamento di
- Personalizzazione delle regole di campionamento
- Opzioni delle regole di campionamento
- Esempi di regole di campionamento
- Configurazione del servizio per l'utilizzo delle regole di campionamento
- Visualizzazione dei risultati del campionamento
- Fasi successive

Configurazione delle regole di campionamento di

Puoi configurare il campionamento per i seguenti casi d'uso:

- API Gateway Entrypoint: API Gateway supporta il campionamento e il tracciamento attivo. Per abilitare il tracciamento attivo su una fase API, consultare Supporto di tracciamento attivo di Amazon API Gateway per AWS X-Ray.
- AWS AppSync— AWS AppSync supporta il campionamento e il tracciamento attivo. Per abilitare il tracciamento attivo sulle AWS AppSync richieste, vedere Tracciamento con AWS X-Ray.
- AWS Step Functions— AWS Step Functions supporta il campionamento e il tracciamento attivo. Per abilitare il tracciamento attivo sulle macchine a AWS Step Functions stati, vedere X-Ray tracing in Step Functions.

- Instrument AWS Distro per OpenTelemetry piattaforme di elaborazione: quando si utilizzano piattaforme di elaborazione come Amazon, EC2 Amazon ECS o AWS Elastic Beanstalk, il campionamento è supportato se l'applicazione è stata strumentata con la versione più recente di AWS Distro for o X-Ray SDK. OpenTelemetry

Personalizzazione delle regole di campionamento

Personalizzando le regole di campionamento, puoi controllare la quantità di dati che registri. È inoltre possibile modificare il comportamento di campionamento senza modificare o ridistribuire il codice. Le regole di campionamento indicano a AWS Distro for OpenTelemetry (ADOT) o X-Ray SDK quante richieste registrare per una serie di criteri. Per impostazione predefinita, l'SDK registra la prima richiesta ogni secondo e il cinque percento di tutte le richieste aggiuntive. Una richiesta al secondo è la riserva. In questo modo viene registrata almeno una traccia al secondo, purché il servizio soddisfi le richieste. Cinque percento è la percentuale di richieste aggiuntive oltre la dimensione della riserva che vengono campionate.

Puoi configurare l'SDK X-Ray per leggere le regole di campionamento da un documento JSON incluso nel codice. Tuttavia, quando esegui più istanze del servizio, ogni istanza esegue il campionamento in modo indipendente. Ciò fa sì che la percentuale globale di richieste campionate aumenti perché tutte le riserve delle istanze sono cumulate insieme in modo efficace. Inoltre, per aggiornare le regole di campionamento locali, è necessario ridistribuire il codice.

Definendo le regole di campionamento nella console X-Ray e configurando l'SDK per leggere le regole dal servizio X-Ray, puoi evitare entrambi questi problemi. Il servizio gestisce la riserva per ciascuna regola e assegna delle quote ad ogni istanza del servizio per distribuire la riserva in modo uniforme, in base al numero di istanze in esecuzione. Il limite del serbatoio viene calcolato in base alle regole impostate. Poiché le regole sono configurate nel servizio, è possibile gestirle senza effettuare implementazioni aggiuntive.

> **Note**
>
> Quando si configurano le regole di campionamento, è fondamentale comprendere che il campionamento a raggi X è «basato sui genitori». Ciò significa che la decisione di campionamento viene presa una sola volta, in genere dal primo X-Ray-enabled servizio che gestisce la richiesta (il servizio «Root»).
> Se un servizio a valle riceve una richiesta che contiene già una decisione di campionamento da parte di un fornitore a monte, rispetterà tale decisione indipendentemente dalle regole di campionamento corrispondenti.

- Quando si applicano le regole: le regole di campionamento personalizzate hanno effetto solo sui servizi per i quali non è stata ancora presa una decisione sul campionamento. Questo di solito si applica a:
  - Il punto di ingresso dell'applicazione (ad esempio, un API Gateway, un Load Balancer o il primo microservizio strumentato).
  - Processi o lavoratori asincroni che avviano una nuova traccia.
- Errore comune: se si crea una regola di campionamento rigorosa per il «Servizio B», ma il «Servizio B» viene sempre chiamato dal «Servizio A», è probabile che la regola per il Servizio B non venga mai applicata perché segue semplicemente la decisione emessa dal Servizio A. Per modificare il campionamento delle tracce per questo flusso di lavoro, è necessario configurare la regola di campionamento sul servizio Root (Servizio A).

> **Note**
>
> X-Ray utilizza un approccio ottimale nell'applicazione delle regole di campionamento e, in alcuni casi, la frequenza di campionamento effettiva potrebbe non corrispondere esattamente alle regole di campionamento configurate. Tuttavia, nel tempo il numero di richieste campionate dovrebbe avvicinarsi alla percentuale configurata.

Ora puoi configurare le regole di campionamento a raggi X direttamente dalla console Amazon. CloudWatch

Per configurare le regole di campionamento nella console CloudWatch

1. Accedi a Console di gestione AWS e apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegli Impostazioni nel riquadro di navigazione a sinistra sotto Configurazione.
3. Scegliete Visualizza impostazioni in Regole di campionamento all'interno della sezione Tracce X-Ray.
4. Per creare una regola, scegliere Create sampling rule (Crea regola di campionamento).

   Per modificare una regola, scegliete una regola e scegliete Modifica per modificarla.

   Per eliminare una regola, scegli una regola e scegli Elimina per eliminarla.

Opzioni delle regole di campionamento

Per ogni regola sono disponibili le seguenti opzioni. I valori di stringa possono utilizzare caratteri jolly per corrispondere a un solo carattere (?) o a zero o a più caratteri (*).

Opzioni delle regole di campionamento

- Nome della regola (stringa): un nome univoco per la regola.
- Priorità (numero intero compreso tra 1 e 9999): la priorità della regola di campionamento. I servizi valutano le regole in ordine crescente di priorità e prendono una decisione sul campionamento in base alla prima regola corrispondente.
- Reservoir (numero intero non negativo): un numero fisso di richieste corrispondenti allo strumento al secondo, prima dell'applicazione del tasso fisso. Il reservoir non viene utilizzato direttamente dai servizi, ma si applica a tutti i servizi che utilizzano la regola nel loro complesso.
- Tasso (numero intero compreso tra 0 e 100): la percentuale di richieste corrispondenti allo strumento, dopo l'esaurimento del serbatoio. Quando configuri una regola di campionamento nella console, scegli una percentuale compresa tra 0 e 100. Quando configuri una regola di campionamento in un SDK client utilizzando un documento JSON, fornisci un valore percentuale compreso tra 0 e 1.
- Nome del servizio (stringa): il nome del servizio strumentato, così come appare nella mappa di traccia.
  - X-Ray SDK: il nome del servizio configurato sul registratore.
  - Amazon API Gateway —*`api-name`*/*`stage`*.
- Tipo di servizio (stringa): il tipo di servizio, così come appare nella mappa di traccia. Per X-Ray SDK, imposta il tipo di servizio applicando il plug-in appropriato:
  - `AWS::ElasticBeanstalk::Environment`— Un AWS Elastic Beanstalk ambiente (plugin).
  - `AWS::EC2::Instance`— Un' EC2 istanza Amazon (plugin).
  - `AWS::ECS::Container`— Un contenitore Amazon ECS (plug-in).
  - `AWS::EKS::Container`— Un contenitore Amazon EKS (plug-in).
  - `AWS::APIGateway::Stage`— Una fase di Amazon API Gateway.
  - `AWS::AppSync::GraphQLAPI` — Una richiesta AWS AppSync API.
  - `AWS::StepFunctions::StateMachine`— Una macchina a AWS Step Functions stati.
- Host (stringa): il nome host dall'intestazione dell'host HTTP.
- Metodo HTTP (stringa) — Il metodo della richiesta HTTP.

- Percorso URL (stringa) — Il percorso URL della richiesta.
  - X-Ray SDK: la parte del percorso dell'URL della richiesta HTTP.
- ARN della risorsa (stringa) — L'ARN della AWS risorsa che esegue il servizio.
  - X-Ray SDK: non supportato. L'SDK può solo utilizzare le regole con Resource ARN (ARN risorse) impostato su `*`.
  - Amazon API Gateway: l'ARN di fase.
- (Facoltativo) Attributi (chiave e valore): attributi del segmento noti al momento della decisione di campionamento.
  - X-Ray SDK: non supportato. L'SDK ignora le regole che specificano degli attributi.
  - Amazon API Gateway: intestazioni della richiesta HTTP originale.
- (Facoltativo) SamplingRateBoost(oggetto): controlla il comportamento di incremento del campionamento basato sulle anomalie.
  - MaxRate (numero intero compreso tra 0 e 100) — La frequenza di campionamento massima (0,0 —1,0) I raggi X possono aumentare fino a durante le anomalie
  - CooldownWindowMinutes (numero intero maggiore di 0) — Intervallo temporale (minuti) in cui è possibile attivare un solo boost, impedendo aumenti continui

Esempi di regole di campionamento

Example— Regola predefinita senza riserva e tariffa bassa

Puoi modificare la riserva e la percentuale della regola di default. La regola di default si applica alle richieste che non soddisfano nessun'altra regola.

- Serbatoio: **`0`**
- Frequenza: **`5`** (**`0.05`**se configurata utilizzando un documento JSON)

Example— Regola di debug per tracciare tutte le richieste relative a un percorso problematico

Un regola ad alta priorità applicata temporaneamente per il debug.

- Nome della regola: **`DEBUG - history updates`**
- Priorità: **`1`**
- Serbatoio: **`1`**
- Frequenza: **`100`** (**`1`**se configurata utilizzando un documento JSON)

- Nome del servizio: **`Scorekeep`**
- Tipo di servizio: **`*`**
- Ospite: **`*`**
- Metodo HTTP: **`PUT`**
- Percorso URL: **`/history/*`**
- ARN della risorsa: **`*`**

Example— Tariffa minima più elevata per POSTs

- Nome della regola: **`POST minimum`**
- Priorità: **`100`**
- Serbatoio: **`10`**
- Frequenza: **`10`** (**`.1`**se configurata utilizzando un documento JSON)
- Nome del servizio: **`*`**
- Tipo di servizio: **`*`**
- Ospite: **`*`**
- Metodo HTTP: **`POST`**
- Percorso URL: **`*`**
- ARN della risorsa: **`*`**

Example Abilita il potenziamento basato sulle anomalie

Configura una regola che attivi un aumento del campionamento fino al 50% durante le anomalie, con un tempo di recupero di 10 minuti.

- Nome della regola: **`MyAdaptiveRule`**
- Priorità: **`100`**
- Serbatoio: **`1`**
- FixedRate: **`0.0510`**
- Nome del servizio: **`*`**
- Tipo di servizio: **`*`**
- Ospite: **`*`**

- Metodo HTTP: **`POST`**
- Percorso URL: **`*`**
- Velocità massima: **`0.5`**
- cooldownWindowMinutes: **`10`**

Configurazione del servizio per l'utilizzo delle regole di campionamento

L'SDK AWS Distro for OpenTelemetry (ADOT) e X-Ray richiedono una configurazione aggiuntiva per utilizzare le regole di campionamento configurate nella console. Consulta la sezione dedicata alla configurazione per il tuo linguaggio per scoprire i dettagli sulla configurazione di una strategia di campionamento:

- Java: utilizzo del campionamento remoto a raggi X con ADOT Java
- Go: configurazione del campionamento con ADOT Go
- Node.js: utilizzo del campionamento remoto a raggi X con ADOT JavaScript
- Python: utilizzo del campionamento remoto a raggi X con ADOT Python
- Rubino: Regole di campionamento
- .NET: utilizzo del campionamento remoto a raggi X con ADOT .NET

Per API Gateway, vedereSupporto di tracciamento attivo di Amazon API Gateway per AWS X-Ray.

Visualizzazione dei risultati del campionamento

La pagina di campionamento della console X-Ray mostra informazioni dettagliate su come i servizi utilizzano ciascuna regola di campionamento.

La colonna Trend (Tendenza) mostra come la regola è stata utilizzata negli ultimi minuti. Ogni colonna mostra le statistiche relative a una finestra di 10 secondi.

Statistiche di campionamento

- Regola di corrispondenza totale: il numero di richieste corrispondenti a questa regola. Questo numero non include le richieste che avrebbero potuto soddisfare questa regola, ma che hanno soddisfatto prima una regola ad alta priorità.
- Totale campionato: il numero di richieste registrate.

- Campionato con tasso fisso: il numero di richieste campionate applicando la tariffa fissa della regola.
- Campionato con limite di riserva: il numero di richieste campionate utilizzando una quota assegnata da X-Ray.
- Prese in prestito dal serbatoio: il numero di richieste campionate prendendo in prestito dal serbatoio. La prima volta che un servizio abbina una richiesta a una regola, X-Ray non gli ha ancora assegnato una quota. Tuttavia, se il serbatoio è almeno 1, il servizio prende in prestito una traccia al secondo fino a quando X-Ray non assegna una quota.

Per ulteriori informazioni sulle statistiche di campionamento e su come i servizi utilizzano le regole di campionamento, consulta Using sampling rules with the X-Ray API (Utilizzo delle regole di campionamento con l'API X-Ray).

Fasi successive

È possibile utilizzare l'API X-Ray per gestire le regole di campionamento. Con l'API puoi creare e aggiornare le regole in modo programmatico in base a una pianificazione o in risposta ad allarmi o notifiche. Consulta Configurazione delle impostazioni di campionamento, gruppi e crittografia con l'API AWS X-Ray per le istruzioni di utilizzo e ulteriori esempi di regole.

AWS Distro for OpenTelemetry, X-Ray SDK Servizi AWS e X-Ray utilizzano anche l'API X-Ray per leggere le regole di campionamento, riportare i risultati del campionamento e ottenere obiettivi di campionamento. I servizi devono tenere traccia della frequenza con cui applicano ogni regola, valutare le regole in base alla priorità e prendere in prestito dal serbatoio quando una richiesta corrisponde a una regola per la quale X-Ray non ha ancora assegnato una quota al servizio. Per ulteriori informazioni su come un servizio utilizza l'API per la campionatura, consulta Using sampling rules with the X-Ray API (Utilizzo delle regole di campionamento con l'API X-Ray).

Quando AWS Distro for OpenTelemetry o X-Ray SDK effettuano il APIs campionamento delle chiamate, usano CloudWatch l'agente come proxy. Se utilizzi già la porta TCP 2000, puoi configurare l'agente per eseguire il proxy su una porta diversa. Per i dettagli, consulta la guida all'installazione dell'CloudWatch agente.

Configurazione del campionamento adattivo

La mancanza di tracce critiche durante i picchi di anomalia può rendere difficile l'analisi della causa principale. Tuttavia, mantenere frequenze di campionamento elevate è costoso. Il campionamento

adattivo a raggi X offre una visibilità completa sulle anomalie e controlla i costi durante le normali operazioni. Con il campionamento adattivo, si imposta una frequenza di campionamento massima e X-Ray si regola automaticamente entro tale limite. X-Ray calcola il boost minimo necessario per catturare le tracce di errore. Se la frequenza di base acquisisce una quantità sufficiente di dati, non si verifica alcun aumento. Paghi solo per il campionamento aggiuntivo quando necessario.

Vantaggi dell'utilizzo del campionamento adattivo:

- Visibilità completa degli incidenti: ottieni tracce complete durante gli incidenti senza intervento manuale. X-Ray regola automaticamente le frequenze di campionamento per catturare le tracce di errore, quindi torna alle frequenze normali.
- Visibilità della causa principale: individua sempre l'origine dei problemi. X-Ray acquisisce dati di errore critici anche quando non viene attivato il campionamento completo delle tracce.
- Ottimizzazione dei costi: brevi incrementi di campionamento (fino a 1 minuto) e periodi di raffreddamento automatici impediscono il sovracampionamento. Paghi solo per i dati necessari per diagnosticare i problemi.

Argomenti

- Supportati SDKs e piattaforme
- Scegli il tuo approccio di campionamento adattivo
- Configurazione SDK locale

Supportati SDKs e piattaforme

SDK supportato: il campionamento adattivo richiede l'SDK AWS Distro for OpenTelemetry (ADOT).

Lingue supportate

- Java (versione v2.11.5 o successiva)
- Python (versione v0.15.0 o successiva)

La tua applicazione deve essere dotata dell'SDK ADOT supportato ed eseguita insieme ad Amazon CloudWatch Agent o Collector. OpenTelemetry

Ad esempio, Amazon EC2, Amazon ECS e Amazon EKS sono piattaforme comuni in cui AWS Application Signals fornisce indicazioni per abilitare l'SDK ADOT e Amazon Agent. CloudWatch

Scegli il tuo approccio di campionamento adattivo

Il campionamento adattivo supporta due approcci, Sampling Boost e Anomaly Span Capture. Questi possono essere applicati indipendentemente o combinati tra loro.

Incremento del campionamento

Adaptive Sampling Boost si basa su regole di campionamento e funziona con il modello di campionamento basato su testine X-Ray esistente. Il campionamento basato sulle teste significa che le decisioni relative al campionamento vengono prese a livello di root service e il flag di campionamento viene trasmesso a valle a tutti i servizi della catena di chiamate.

- Potenziamento basato su regole: il potenziamento è sempre legato a una regola di campionamento a raggi X specifica. Ogni regola può definire la propria frequenza di incremento massima e il proprio comportamento di raffreddamento.
- Campionamento basato sulla testa: le decisioni relative al campionamento vengono prese presso il servizio root e il flag di campionamento viene passato a valle a tutti i servizi della catena di chiamate.
- Basato sulle anomalie: X-Ray si affida all'SDK per riportare le statistiche sulle anomalie. Quando X-Ray rileva anomalie come errori o latenza elevata, utilizza queste statistiche per calcolare una frequenza di incremento appropriata (fino al massimo configurato).

Segnalazione delle anomalie

Ogni servizio applicativo nella catena di chiamate può emettere statistiche sulle anomalie tramite l'SDK richiesto:

- Servizio root: deve essere eseguito su un SDK e una piattaforma supportati per abilitare il sampling boost. Se il servizio root non è supportato, non si verificherà alcun boost.
- Servizi downstream: i servizi downstream segnalano solo le anomalie e non possono prendere decisioni di campionamento. Quando un servizio downstream esegue un SDK supportato, le anomalie rilevate possono innescare un aumento del campionamento. Quando un servizio downstream non è supportato (ad esempio, se si esegue un SDK precedente), le anomalie su quel servizio non generano alcun incremento. Questi servizi possono comunque propagare il contesto a valle se seguono la propagazione del contesto standard (come W3C trace context e baggage). Ciò garantisce che i servizi supportati SDKs in ulteriori servizi a valle possano segnalare anomalie che generano un aumento.

Aumenta i tempi e la portata

- Ritardo di attivazione: è possibile aspettarsi che un aumento del campionamento inizi già 10 secondi dopo che X-Ray rileva un'anomalia.
- Periodo di potenziamento: dopo che X-Ray ha attivato un boost, dura fino a 1 minuto prima di tornare alla frequenza di campionamento di base.
- Migliora il raffreddamento: dopo che si è verificato un potenziamento, X-Ray non attiverà un altro potenziamento per la stessa regola fino al termine della finestra di raffreddamento.

  Ad esempio, se si imposta su `cooldown` 10 minuti, una volta terminato un boost, non è possibile attivare un nuovo boost fino alla finestra di 10 minuti successiva.

  Caso speciale: se si imposta su `cooldown` 1 minuto e poiché un boost stesso può durare fino a 1 minuto, i potenziamenti possono essere attivati in modo efficace e continuo se l'anomalia persiste.

> **Note**
>
> Usa piattaforme supportate SDKs e per il tuo root service. Sampling boost funziona solo con piattaforme SDKs supportate. Sebbene sampling boost abbia un'alta probabilità di catturare tracce di anomalia, potrebbe non catturare tutte le tracce di anomalia.

Aumenta la visibilità

Quando una regola di campionamento è configurata con Adaptive Sampling Boost, X-Ray emette automaticamente metriche vended che consentono di monitorare l'attività di boost.

- Nome della metrica: `SamplingRate`
- Dimensione — `RuleName` (impostata sul nome effettivo della regola)

Ogni regola `SamplingRateBoost` abilitata pubblicherà la frequenza di campionamento effettiva, inclusa la frequenza di base e gli eventuali aumenti temporanei. Ciò consente di:

- Tieni traccia di quando vengono attivati i potenziamenti
- Monitora la frequenza di campionamento effettiva per ogni regola
- Correla gli incrementi con le anomalie delle applicazioni (ad esempio picchi di errore o eventi di latenza)

Puoi visualizzare questi parametri in Amazon CloudWatch Metrics, nello spazio dei nomi AWS/X-Ray. Il valore della metrica è un numero a virgola mobile compreso tra 0 e 1, che rappresenta la frequenza di campionamento effettiva.

Configurare l'incremento del campionamento utilizzando le regole di campionamento a raggi X

È possibile abilitare il campionamento adattivo direttamente nelle regole di campionamento a raggi X esistenti aggiungendo un nuovo campo. `SamplingRateBoost` Per ulteriori informazioni, vedere Personalizzazione delle regole di campionamento. Ciò fornisce un modo centralizzato per abilitare il campionamento adattivo senza modificare il codice dell'applicazione o applicare la distribuzione dell'applicazione. Quando abiliti il campionamento adattivo, X-Ray aumenta automaticamente il campionamento in caso di anomalie come picchi di errore o valori anomali di latenza, mantenendo al contempo le frequenze di campionamento entro il massimo configurato. `SamplingRateBoost`può essere applicato a qualsiasi regola di campionamento personalizzata tranne la regola di campionamento. `Default`

Il `SamplingRateBoost` campo definisce il limite superiore e il comportamento per il campionamento basato sulle anomalie.

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

`MaxRate`Definisce la frequenza di campionamento massima che X-Ray applicherà quando rileva anomalie. L'intervallo di valori è pari a. `0.0` `1.0` Ad esempio, `"MaxRate": 0.25` consente di aumentare il campionamento fino al 25% delle richieste durante una finestra di anomalia. X-Ray determina la frequenza appropriata tra il valore di base e il valore massimo, a seconda dell'attività dell'anomalia.

La finestra temporale `CooldownWindowMinutes` definisce (in minuti) in cui è possibile attivare un solo aumento della frequenza di campionamento. Dopo che si è verificato un aumento, non sono consentiti ulteriori incrementi fino alla finestra successiva. Il tipo di valore è intero (minuti).

`DisableDefaultAnomalyDetection`Controlla se l'ADOT SDK utilizza la condizione di anomalia predefinita integrata (risposte di errore HTTP 5xx) per attivare un aumento del campionamento. Il tipo

di valore è booleano e il valore predefinito è. `false` Se impostato su`true`, l'SDK non aumenterà più il campionamento sulle risposte HTTP 5xx per impostazione predefinita. In tal caso, un boost viene attivato solo da condizioni di anomalia definite esplicitamente tramite. Configurazione SDK locale Utilizzalo quando desideri il pieno controllo su ciò che costituisce un'anomalia (ad esempio, solo codici di errore specifici o soglie di latenza) e vuoi evitare aumenti su tutte le risposte 5xx.

Regola di esempio con campionamento adattivo

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

In questo esempio, il campionamento di base è del 5% (). `FixedRate: 0.05` Durante le anomalie, X-Ray può aumentare il campionamento fino al 25% (). `MaxRate: 0.25` Potenzia solo una volta ogni 10 minuti.

Configurazione delle condizioni di anomalia

Quando non viene fornita alcuna configurazione della condizione di anomalia, l'ADOT SDK utilizza i codici di errore HTTP 5xx come condizione di anomalia predefinita per attivare l'incremento del campionamento. È possibile disabilitare questa impostazione predefinita impostandola sul campo. `DisableDefaultAnomalyDetection` `true` `SamplingRateBoost`

È inoltre possibile ottimizzare le condizioni di anomalia localmente nell'SDK ADOT supportato utilizzando variabili di ambiente. Per ulteriori informazioni, consulta Configurazione SDK locale.

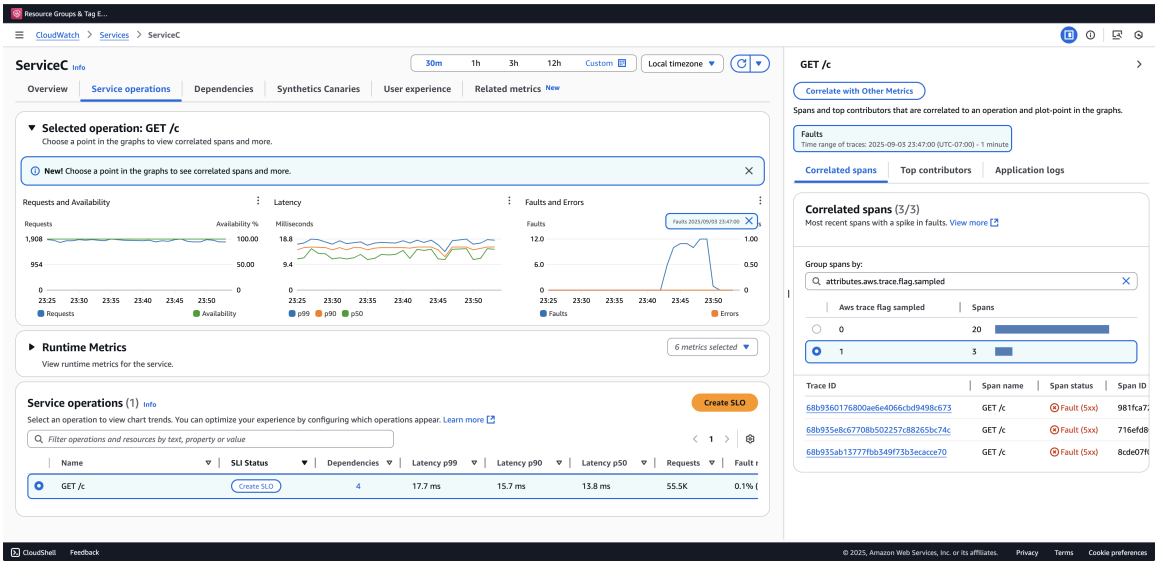
L'anomalia si estende all'acquisizione

L'acquisizione dell'intervallo di anomalie assicura che gli intervalli critici che rappresentano le anomalie vengano sempre registrati, anche se non viene campionata la traccia completa. Questa funzionalità integra l'incremento del campionamento concentrandosi sull'acquisizione dell'anomalia stessa, piuttosto che sull'aumento del campionamento per tracce future.

Quando l'SDK ADOT rileva un'anomalia, emette immediatamente tale intervallo, indipendentemente dalla decisione di campionamento. Poiché l'SDK emette solo intervalli relativi all'anomalia, queste tracce sono tracce parziali, non transazioni complete. end-to-end

Una volta che l'SDK ADOT rileva un intervallo di anomalia, tenta di emettere il maggior numero possibile di intervalli dalla stessa traccia. Tutti gli intervalli emessi con questa funzione sono contrassegnati con l'attributo,. `aws.trace.flag.sampled = 0` Ciò consente di distinguere facilmente le tracce parziali (acquisizione delle anomalie) dalle tracce complete (campionamento normale) nella ricerca e nell'analisi delle transazioni.

Consigliamo di attivare Transaction Search per visualizzare e interrogare le tracce parziali. L'esempio seguente mostra una pagina di servizio nella console Application Signals. ServiceC è configurato con anomaly span capture e fa parte di una catena di chiamate in cui si applica il sampling boost. Queste configurazioni generano tracce complete e parziali. È possibile utilizzare l'`aws.trace.flag.sampled`attributo per distinguere i tipi di tracce.



L'acquisizione degli intervalli di anomalia può essere abilitata o personalizzata solo tramite. Configurazione SDK locale

Configurazione SDK locale

È possibile configurare le funzionalità di campionamento adattivo in ADOT SDK fornendo una configurazione YAML tramite una variabile di ambiente. La configurazione locale fornisce un controllo dettagliato sulle condizioni di anomalia e sulle soglie.

Ciò è necessario per l'acquisizione dell'intervallo di anomalie e facoltativo per personalizzare le condizioni di aumento del campionamento. Di seguito è riportato un esempio di configurazione:

```
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
```

Le definizioni dei campi sono riportate di seguito:

- `version`— Versione dello schema per il file di configurazione
- `anomalyConditions`— Definisce le condizioni in cui vengono rilevate le anomalie e il modo in cui vengono utilizzate
  - `errorCodeRegex`— Espressione regolare che definisce quali codici di stato HTTP sono considerati anomalie
  - `operations`— Elenco delle operazioni o degli endpoint a cui si applica la condizione
  - `highLatencyMs`— Soglia di latenza (in millisecondi) al di sopra della quale gli intervalli vengono considerati anomalie
  - `usage`— Definisce a quale funzionalità si applica la condizione:
    - `both`— Si applica al sampling boost e all'anomaly span capture (impostazione predefinita se l'utilizzo non è specificato)

  - `sampling-boost`— Utilizzato solo per attivare gli incrementi di campionamento
  - `anomaly-span-capture`— Utilizzato solo per l'acquisizione dell'intervallo di anomalie
- `anomalyCaptureLimit`— Definisce i limiti al numero di tracce con intervalli di anomalia emesse.

  `anomalyTracesPerSecond`— Numero massimo di tracce con intervalli di anomalia acquisiti al secondo, per evitare un volume di intervallo eccessivo (il valore predefinito è 1 se non è presente). anomalyCaptureLimit

> **Note**
>
> - `AnomalyConditions`sostituisce la condizione di anomalia predefinita per sampling boost (HTTP 5xx). Se si desidera mantenere la condizione predefinita durante l'utilizzo della configurazione locale, è necessario includerla esplicitamente in qualsiasi elemento di. `AnomalyConditions`
> - Per ogni `anomalyConditions` articolo:
>   - Quando il `operations` campo viene omesso, la condizione si applica a tutte le operazioni (livello di servizio)
>   - Quando il `operations` campo è presente ma impostato su un elenco vuoto, la condizione non si applica a nessuna operazione, rendendo tale elemento un no-op
>   - Quando entrambe `errorCodeRegex` le risposte `highLatencyMs` vengono omesse, la condizione non ha criteri di anomalia da valutare, il che rende tale elemento una condizione non operativa
> - Relazioni logiche:
>   - Tra gli elementi in`anomalyConditions`, la relazione è OR.
>   - All'interno di un singolo elemento, più campi (ad esempio `errorCodeRegex` e`highLatencyMs`) vengono combinati con AND.
>
>   Esempio:
>
>     ```
>     errorCodeRegex: "^429|5\\d\\d$"
>     highLatencyMs: 300
>     ```

Questa condizione indica che il codice di stato corrisponde a 429 o 5xx AND ha una latenza ≥ 300 ms.

Applica la configurazione locale a ADOT SDK

È possibile applicare la configurazione locale all'ADOT SDK impostando la variabile di ambiente. `AWS_XRAY_ADAPTIVE_SAMPLING_CONFIG` Il valore deve essere un documento YAML valido (in linea o annidato).

Ad esempio, Amazon EC2 e Amazon ECS, impostano direttamente la variabile di ambiente:

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

Per Amazon EKS, definisci la variabile di ambiente all'interno delle specifiche del pod come YAML annidato:

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

Deep linking tra console

Puoi utilizzare percorsi e query per creare collegamenti diretti a tracce specifiche o visualizzazioni filtrate delle tracce e della mappa di traccia.

Pagine della console

- Pagina di benvenuto — xray/home#/welcome
- Guida introduttiva — xray/home#/getting-Iniziata
- Mappa di tracciamento — xray/home#/service -map
- Tracce — xray/home#/traces

Tracce

Puoi generare link per la sequenza temporale, i dati non elaborati e le visualizzazioni sulla mappa di specifici tracciamenti.

Traccia la cronologia — `xray/home#/traces/`*`trace-id`*

Dati di tracciamento grezzi — `xray/home#/traces/`*`trace-id`*`/raw`

Example— dati di traccia non elaborati

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

Espressioni filtro

Collegamento a un elenco filtrato di tracciamenti.

Visualizzazione filtrata delle tracce — `xray/home#/traces?filter=`*`filter-expression`*

Example— espressione di 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— espressione di filtro (URL codificato)

```
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
```

Per ulteriori informazioni sulle espressioni filtro, consulta Utilizzo delle espressioni di filtro.

Intervallo temporale

Specificare un periodo di tempo o l'ora di inizio e di fine nel formato ISO86 01. Gli intervalli di tempo sono espressi in UTC e possono durare fino a 6 ore.

Periodo di tempo — `xray/home#/`*`page`*`?timeRange=`*`range-in-minutes`*

Example— mappa di tracciamento dell'ultima ora

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

Ora di inizio e fine — `xray/home#/`*`page`*`?timeRange=`*`start`*`~`*`end`*

Example— intervallo di tempo preciso in secondi

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

Example— intervallo di tempo preciso in minuti

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

Regione

Specificate un link Regione AWS alle pagine di quella regione. Se non si specifica una regione, la console reindirizza l'utente all'ultima regione visitata.

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

Example— mappa di tracciamento negli Stati Uniti occidentali (Oregon) (us-west-2)

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

Quando includi una regione con altri parametri di query, la query Region precede l'hash e le X-Ray-specific query seguono il nome della pagina.

Example— tracciamento della mappa dell'ultima ora negli Stati Uniti occidentali (Oregon) (us-west-2)

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

Combinazioni

Example— tracce recenti con un filtro di durata

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

Output

- Pagina — Tracce
- Intervallo di tempo: ultimi 15 minuti
- Filtro: durata >= 5 E durata <= 8

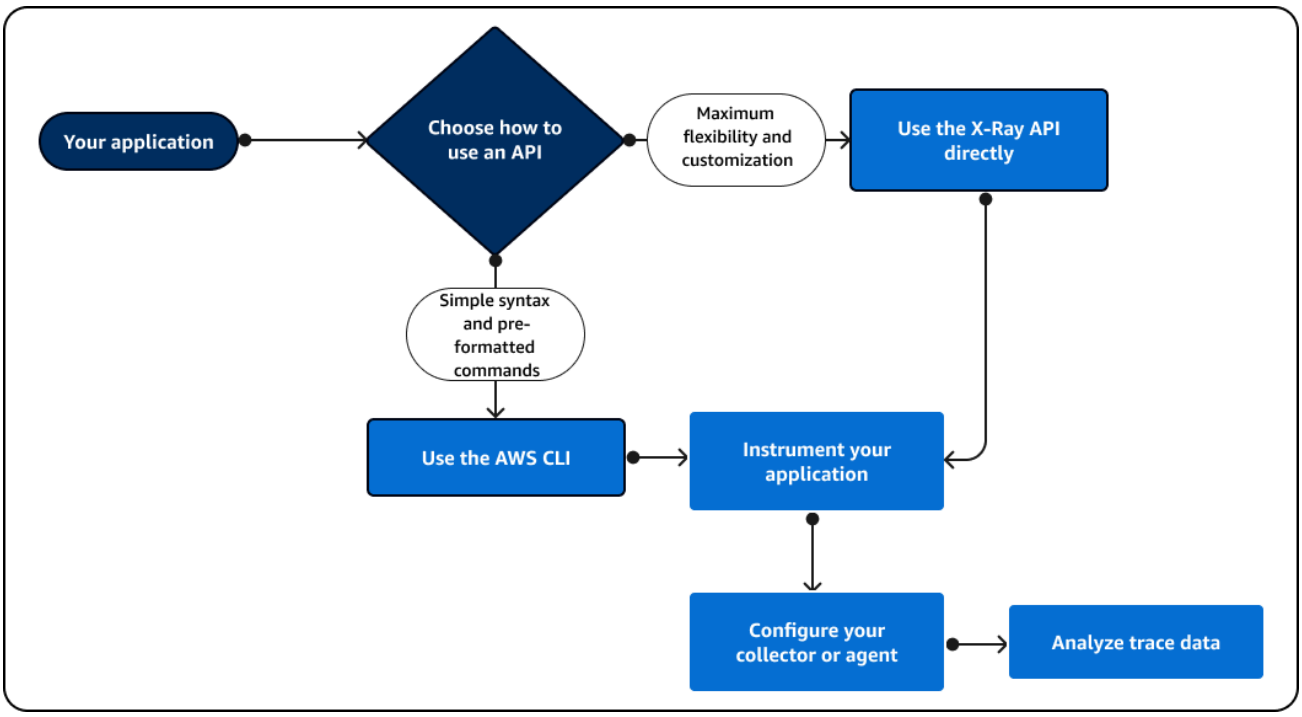
Usa l' X-Ray API

Se l' X-Ray SDK non supporta il tuo linguaggio di programmazione, puoi utilizzare direttamente le X-Ray API o AWS Command Line Interface (AWS CLI) per chiamare i comandi X-Ray API. Usa le seguenti linee guida per scegliere come interagire con l'API:

- Utilizza AWS CLI la sintassi più semplice utilizzando comandi preformattati o con opzioni all'interno della richiesta.
- Usa direttamente l' X-Ray API per la massima flessibilità e personalizzazione delle richieste che invii. X-Ray

Se si utilizza direttamente l'X-Ray API anziché il AWS CLI, è necessario parametrizzare la richiesta nel formato di dati corretto e potrebbe anche essere necessario configurare l'autenticazione e la gestione degli errori.

Il diagramma seguente mostra le linee guida per scegliere come interagire con l' X-RayAPI:



Utilizza l' X-Ray API per inviare i dati di tracciamento direttamente a X-Ray. L' X-Ray API supporta tutte le funzioni disponibili nell' X-Ray SDK, incluse le seguenti azioni comuni:

- PutTraceSegments— Carica i documenti dei segmenti in. X-Ray
- BatchGetTraces— Recupera un elenco di tracce in un elenco di ID di traccia. Ogni traccia recuperata è una raccolta di documenti segmentati da una singola richiesta.
- GetTraceSummaries— Recupera gli ID e le annotazioni per le tracce. È possibile specificare `FilterExpression` a per recuperare un sottoinsieme di riepiloghi delle tracce.
- GetTraceGraph— Recupera un grafico del servizio per un ID di traccia specifico.
- GetServiceGraph— Recupera un documento JSON formattato che descrive i servizi che elaborano le richieste in arrivo e chiamano le richieste a valle.

È inoltre possibile utilizzare la AWS Command Line Interface (AWS CLI) all'interno del codice dell'applicazione per interagire in modo programmatico. X-Ray AWS CLI Supporta tutte le funzioni disponibili nell' X-Ray SDK, incluse quelle per altri. Servizi AWS Le seguenti funzioni sono versioni delle operazioni API elencate in precedenza con un formato più semplice:

- put-trace-segments — Carica i documenti dei segmenti in. X-Ray
- batch-get-trace — Recupera un elenco di tracce in un elenco di ID di traccia. Ogni traccia recuperata è una raccolta di documenti segmentati da una singola richiesta.

- get-trace-summaries: recupera gli ID e le annotazioni per le tracce. È possibile specificare a per recuperare un sottoinsieme di riepiloghi `FilterExpression` delle tracce.
- get-trace-graph — Recupera un grafico del servizio per un ID di traccia specifico.
- get-service-graph — Recupera un documento `JSON` formattato che descrive i servizi che elaborano le richieste in arrivo e chiamano le richieste a valle.

Per iniziare, devi installare il file per il tuo sistema operativo. AWS CLI AWS supporti Linux macOS e sistemi Windows operativi. Per ulteriori informazioni sull'elenco dei X-Ray comandi, consulta la guida di riferimento ai AWS CLI comandi per X-Ray.

Argomenti

- Utilizzo dell' AWS X-Ray API con la AWS CLI
- Invio di dati di tracciamento a AWS X-Ray
- Ottenere dati da AWS X-Ray
- Configurazione delle impostazioni di campionamento, gruppi e crittografia con l'API AWS X-Ray
- Using sampling rules with the X-Ray API (Utilizzo delle regole di campionamento con l'API X-Ray)
- AWS X-Ray segmentare i documenti

Utilizzo dell' AWS X-Ray API con la AWS CLI

La AWS CLI consente di accedere direttamente al servizio X-Ray e di utilizzare lo stesso che la console X-Ray utilizza per recuperare APIs il grafico del servizio e i dati grezzi delle tracce. L'applicazione di esempio include script che mostrano come utilizzarli APIs con la AWS CLI.

Prerequisiti

Questo tutorial utilizza l'applicazione di esempio Scorekeep e gli script in essa inclusi per generare dei dati di tracciamento e una mappa del servizio. Segui le istruzioni nel tutorial sulle nozioni di base per avviare l'applicazione.

Questo tutorial utilizza AWS CLI per mostrare l'uso di base dell'API X-Ray. La AWS CLI, disponibile per Windows, Linux e OS-X, fornisce l'accesso da riga di comando al pubblico per tutti. APIs Servizi AWS

> **Note**
>
> È necessario verificare che sia configurato nella stessa regione in cui AWS CLI è stata creata l'applicazione di esempio Scorekeep.

Gli script incluso per testare l'applicazione di esempio utilizzano `cURL` per inviare il traffico verso l'API e `jq` per analizzare l'output. Puoi scaricare l'eseguibile di `jq` da stedolan.github.io e l'eseguibile di `curl` da https://curl.haxx.se/download.html. La maggior parte delle installazioni Linux e OS X include cURL.

Generare i dati di tracciamento

Mentre il gioco è in corso, l'applicazione web genera continuamente traffico verso l'API ogni pochi secondi, ma genera solo un tipo di richiesta. Utilizza lo script `test-api.sh` per eseguire scenari completi e generare dati di tracciamento più eterogeni mentre testi l'API.

Per usare lo script **`test-api.sh`**

1. Apri la console Elastic Beanstalk.
2. Accedi alla console di gestione del tuo ambiente.
3. Copiare l'URL di ambiente dall'intestazione della pagina.
4. Aprire `bin/test-api.sh` e sostituire il valore per l'API con l'URL del proprio ambiente.

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

5. Eseguire lo script per generare il traffico verso l'API.

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

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

Usa l'API X-Ray

La AWS CLI fornisce comandi per tutte le azioni API fornite da X-Ray, tra cui e. `GetServiceGraph` `GetTraceSummaries` Per ulteriori informazioni su tutte le operazioni supportate e i tipi di dati che utilizzano, consulta la Guida di riferimento delle API di AWS X-Ray.

Example bin/service-graph.sh

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

Lo script recupera un grafo del servizio relativo agli ultimi 10 minuti.

```
~/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'
```

Lo script recupera le URLs tracce generate tra uno e due minuti fa.

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

Lo script recupera i tracciamenti completi generati tra uno e due minuti fa.

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

Rimozione

Termina il tuo ambiente Elastic Beanstalk per chiudere le istanze EC2 Amazon, le tabelle DynamoDB e altre risorse.

Per terminare l'ambiente Elastic Beanstalk

1. Apri la console Elastic Beanstalk.

2. Accedi alla console di gestione del tuo ambiente.
3. Scegliere Actions (Operazioni).
4. Scegliere Terminate Environment (Termina ambiente).
5. Scegliere Terminate (Termina).

I dati di traccia vengono eliminati automaticamente da X-Ray dopo 30 giorni.

Invio di dati di tracciamento a AWS X-Ray

È possibile inviare dati di traccia a X-Ray sotto forma di documenti di segmento. Un documento di segmento è una stringa in formato JSON che contiene le informazioni sull'attività eseguita dalla tua applicazione per il soddisfacimento di una richiesta. La tua applicazione può memorizzare i dati sull'attività che svolge in segmenti, oppure le attività che utilizzano i servizi e le risorse a valle possono creare dei sottosegmenti.

I segmenti registrano le informazioni sull'attività svolta dalla tua applicazione. Un segmento registra almeno il tempo impiegato per un'attività, un nome e due IDs. L'ID di tracciamento monitora la richiesta durante il suo passaggio tra i servizi. L'ID segmento monitora l'attività svolta per la richiesta proveniente da uno specifico servizio.

Example Segmento completo minimale

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

Quando ricevi una richiesta, puoi inviare un segmento in esecuzione come segnaposto fino al completamento della richiesta.

Example Segmento in esecuzione

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

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

È possibile inviare segmenti a X-Ray direttamente, `PutTraceSegments`con o tramite il demone X-Ray.

La maggior parte delle applicazioni richiama altri servizi o accede a risorse con l'SDK. AWS Registra le informazioni sulle chiamate downstream in sottosegmenti. X-Ray utilizza i sottosegmenti per identificare i servizi a valle che non inviano segmenti e creare le relative voci nel grafico dei servizi.

Un sottosegmento può essere incorporato in un documento di segmento completo oppure può essere inviato in modo separato. Invia i sottosegmenti separatamente per tracciare in modo asincrono le chiamate downstream per richieste di lunga durata o per evitare di superare la dimensione massima del documento del segmento (64 kB).

Example Segmento secondario

Un sottosegmento presenta un `type` valorizzato a `subsegment` e un `parent_id` che identifica il segmento padre.

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

Per ulteriori informazioni sui campi e i valori che puoi includere in segmenti e sottosegmenti, consulta AWS X-Ray segmentare i documenti.

Sections

- Generazione di tracce IDs
- Usando PutTraceSegments
- Invio di documenti di segmento al demone X-Ray

Generazione di tracce IDs

Per inviare dati a X-Ray, è necessario generare un ID di traccia univoco per ogni richiesta.

Formato ID di traccia X-Ray

Una radiografia `trace_id` è composta da tre numeri separati da trattini. Ad esempio, `1-58406520-a006649127e371903a2de979`. Questo include:

- Il numero di versione, che è. 1
- L'ora della richiesta originale in Unix epoch time utilizzando 8 cifre esadecimali.

  Ad esempio, il fuso orario PST delle 10:00 del 1° dicembre 2016 è espresso in secondi o in cifre esadecimali. `1480615200` `58406520`
- Un identificatore a 96 bit univoco a livello globale per la traccia in 24 cifre esadecimali.

> **Note**
>
> X-Ray ora supporta IDs le tracce create utilizzando OpenTelemetry e qualsiasi altro framework conforme alla specifica W3C Trace Context. Un ID di traccia W3C deve essere formattato nel formato X-Ray Trace ID quando viene inviato a X-Ray. Ad esempio, l'ID di traccia W3C `4efaaf4d1e8720b39541901950019ee5` deve essere formattato come `1-4efaaf4d-1e8720b39541901950019ee5` quando si invia a X-Ray. La traccia X-Ray IDs include il timestamp originale della richiesta in Unix epoch Time, ma questo non è necessario quando si invia la traccia W3C in formato X-Ray. IDs

È possibile scrivere uno script per generare una traccia a raggi X IDs per il test. Qui di seguito sono riportati due esempi.

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"
```

Vedi l'applicazione di esempio Scorekeep per gli script che creano, tracciano IDs e inviano segmenti al demone X-Ray.

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

Usando PutTraceSegments

Puoi caricare i documenti di segmento tramite l'API `PutTraceSegments`. L'API dispone di un singolo parametro, `TraceSegmentDocuments`, che richiede un elenco di documenti di segmento in formato JSON.

Con l'AWS CLI, usa il `aws xray put-trace-segments` comando per inviare documenti di segmento direttamente a X-Ray.

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

> **Note**
>
> Windows Command Processor e Windows PowerShell hanno requisiti diversi per la citazione e l'escape delle virgolette nelle stringhe JSON. Per i dettagli, consulta Utilizzo delle virgolette nelle stringhe nella Guida per l'utente della AWS CLI .

L'output elenca tutti i segmenti per i quali l'elaborazione non è riuscita. Ad esempio, se la data nell'ID di tracciamento è troppo remota nel passato, verrà visualizzato un errore come il seguente.

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

Puoi passare più documenti di segmento insieme, separati da spazi.

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

Invio di documenti di segmento al demone X-Ray

Invece di inviare documenti di segmento all'API X-Ray, puoi inviare segmenti e sottosegmenti al daemon X-Ray, che li memorizzerà nel buffer e li caricherà sull'API X-Ray in batch. L'X-Ray SDK invia i documenti dei segmenti al demone per evitare di effettuare chiamate dirette. AWS

> **Note**
>
> Per istruzioni sull'esecuzione del daemon, consulta Esecuzione locale del demone X-Ray.

Invia il segmento in formato JSON tramite la porta UDP 2000, anteponendo l'intestazione per il 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"}
```

Su Linux, puoi inviare documenti di segmento al daemon tramite un terminale Bash. Salva l'intestazione e il documento di segmento in un file di testo e reindirizzalo verso `/dev/udp` con `cat`.

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

Example segment.txt

```
{"format": "json", "version": 1}
```

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

Controllate il registro del demone per verificare che abbia inviato il segmento a X-Ray.

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

Ottenere dati da AWS X-Ray

AWS X-Ray elabora i dati di traccia che gli invii per generare tracce complete, riepiloghi di traccia e grafici di servizio in JSON. Puoi recuperare i dati generati direttamente dall'API con la AWS CLI.

Sections

- Recupero del grafico del servizio
- Recupero del grafico del servizio dal gruppo
- Recupero dei tracciamenti
- Recupero e definizione dell'analisi delle cause principali

Recupero del grafico del servizio

Puoi utilizzare l'API `GetServiceGraph` per recuperare il grafico del servizio in formato JSON. L'API richiede un'ora di inizio e di fine, che puoi calcolare in un terminale Linux con il comando `date`.

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

`date +%s` converte una data in secondi. Utilizza questo numero come ora finale e sottrai da esso il tempo necessario per ottenere un'ora di inizio.

Example Script per recuperare un grafico del servizio relativo agli ultimi 10 minuti

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

L'esempio seguente mostra un grafico di servizio con 4 nodi, tra cui un nodo client, un' EC2 istanza, una tabella DynamoDB e un argomento Amazon SNS.

Example GetServiceGraph output


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

Recupero del grafico del servizio dal gruppo

Per effettuare la chiamata a un grafico del servizio in base al contenuto di un gruppo, includi `groupName` o `groupARN`. L'esempio seguente mostra una chiamata del grafico del servizio a un gruppo denominato Example1.

Example Script per recuperare un grafico del servizio in base al nome del gruppo Example1

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

Recupero dei tracciamenti

Puoi utilizzare l'API `GetTraceSummaries` per ottenere un elenco di riepiloghi dei tracciamenti. I riepiloghi delle tracce includono informazioni che è possibile utilizzare per identificare le tracce che si desidera scaricare per intero, tra cui annotazioni, informazioni su richieste e risposte e. IDs

Ci sono due flag `TimeRangeType` disponibili quando si chiama `aws xray get-trace-summaries`:

- TraceId— La `GetTraceSummaries` ricerca predefinita utilizza l'ora TraceID e restituisce le tracce iniziate all'interno dell'intervallo calcolato. `[start_time, end_time)` Questo intervallo di timestamp viene calcolato in base alla codifica del timestamp all'interno di o può essere definito manualmente. TraceId
- Ora dell'evento: per cercare gli eventi man mano che si verificano nel tempo, AWS X-Ray consente di cercare le tracce utilizzando i timestamp degli eventi. L'ora dell'evento restituisce i tracciamenti attivi durante l'intervallo `[start_time, end_time)`, indipendentemente dall'inizio del tracciamento.

Utilizza il comando `aws xray get-trace-summaries` per ottenere un elenco di riepiloghi di tracciamento. I seguenti comandi ottengono un elenco di riepiloghi delle tracce relative a un periodo compreso tra 1 e 2 minuti nel passato utilizzando l'ora predefinita. TraceId

Example Script per ottenere i riepiloghi dei tracciamenti

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

Example GetTraceSummaries uscita

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

```
                    "UserName": "5M388M1E"
                }
            ],
            "HasFault": false,
            "Annotations": {
                "UserID": [
                    {
                        "AnnotationValue": {
                            "StringValue": "5M388M1E"
                        }
                    }
                ],
                "Name": [
                    {
                        "AnnotationValue": {
                            "StringValue": "Ola"
                        }
                    }
                ]
            },
            "ResponseTime": 3.232,
            "Duration": 3.232,
            "Id": "1-59602603-23fc5b688855d396af79b496",
            "HasThrottle": false
        }
    ],
    "ApproximateTime": 1499473304.0,
    "TracesProcessedCount": 2
}
```

Utilizza l'ID di tracciamento nell'output per recuperare un tracciamento completo con l'API `BatchGetTraces`.

Example BatchGetTraces comando

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

Example BatchGetTraces uscita

```
{
    "Traces": [
        {
            "Duration": 3.232,
```

```
            "Segments": [
                {
                    "Document": "{\"id\":\"1fb07842d944e714\",\"name\":
\"random-name\",\"start_time\":1.499473411677E9,\"end_time\":1.499473414572E9,
\"parent_id\":\"0c544c1b1bbff948\",\"http\":{\"response\":{\"status\":200}},
\"aws\":{\"request_id\":\"ac086670-6373-11e7-a174-f31b3397f190\"},\"trace_id\":
\"1-59602603-23fc5b688855d396af79b496\",\"origin\":\"AWS::Lambda\",\"resource_arn\":
\"arn:aws:lambda:us-west-2:123456789012:function:random-name\"}",
                    "Id": "1fb07842d944e714"
                },
                {
                    "Document": "{\"id\":\"194fcc8747581230\",\"name\":\"Scorekeep
\",\"start_time\":1.499473411562E9,\"end_time\":1.499473414794E9,\"http\":{\"request
\":{\"url\":\"http://scorekeep.elasticbeanstalk.com/api/user\",\"method\":\"POST\",
\"user_agent\":\"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/59.0.3071.115 Safari/537.36\",\"client_ip\":\"205.251.233.183\"},
\"response\":{\"status\":200}},\"aws\":{\"elastic_beanstalk\":{\"version_label\":\"app-
abb9-170708_002045\",\"deployment_id\":406,\"environment_name\":\"scorekeep-dev\"},
\"ec2\":{\"availability_zone\":\"us-west-2c\",\"instance_id\":\"i-0cd9e448944061b4a
\"},\"xray\":{\"sdk_version\":\"1.1.2\",\"sdk\":\"X-Ray for Java\"}},\"service
\":{},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"user\":\"5M388M1E
\",\"origin\":\"AWS::ElasticBeanstalk::Environment\",\"subsegments\":[{\"id\":
\"0c544c1b1bbff948\",\"name\":\"Lambda\",\"start_time\":1.499473411629E9,\"end_time
\":1.499473414572E9,\"http\":{\"response\":{\"status\":200,\"content_length\":14}},
\"aws\":{\"log_type\":\"None\",\"status_code\":200,\"function_name\":\"random-name
\",\"invocation_type\":\"RequestResponse\",\"operation\":\"Invoke\",\"request_id
\":\"ac086670-6373-11e7-a174-f31b3397f190\",\"resource_names\":[\"random-name\"]},
\"namespace\":\"aws\"},{\"id\":\"071684f2e555e571\",\"name\":\"## UserModel.saveUser
\",\"start_time\":1.499473414581E9,\"end_time\":1.499473414769E9,\"metadata\":{\"debug
\":{\"test\":\"Metadata string from UserModel.saveUser\"}},\"subsegments\":[{\"id\":
\"4cd3f10b76c624b4\",\"name\":\"DynamoDB\",\"start_time\":1.49947341469E9,\"end_time
\":1.499473414769E9,\"http\":{\"response\":{\"status\":200,\"content_length\":57}},
\"aws\":{\"table_name\":\"scorekeep-user\",\"operation\":\"UpdateItem\",\"request_id
\":\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738BOB4KQNSO5AEMVJF66Q9\",\"resource_names\":
[\"scorekeep-user\"]},\"namespace\":\"aws\"}]}]}",
                    "Id": "194fcc8747581230"
                },
                {
                    "Document": "{\"id\":\"00f91aa01f4984fd\",\"name\":
\"random-name\",\"start_time\":1.49947341283E9,\"end_time\":1.49947341457E9,
\"parent_id\":\"1fb07842d944e714\",\"aws\":{\"function_arn\":\"arn:aws:lambda:us-
west-2:123456789012:function:random-name\",\"resource_names\":[\"random-name\"],
\"account_id\":\"123456789012\"},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",
\"origin\":\"AWS::Lambda::Function\",\"subsegments\":[{\"id\":\"e6d2fe619f827804\",
```

```
\"name\":\"annotations\",\"start_time\":1.499473413012E9,\"end_time\":1.499473413069E9,
\"annotations\":{\"UserID\":\"5M388M1E\",\"Name\":\"Ola\"}},{\"id\":\"b29b548af4d54a0f
\",\"name\":\"SNS\",\"start_time\":1.499473413112E9,\"end_time\":1.499473414071E9,
\"http\":{\"response\":{\"status\":200}},\"aws\":{\"operation\":\"Publish\",
\"region\":\"us-west-2\",\"request_id\":\"a2137970-f6fc-5029-83e8-28aadeb99198\",
\"retries\":0,\"topic_arn\":\"arn:aws:sns:us-west-2:123456789012:awseb-e-
ruag3jyweb-stack-NotificationTopic-6B829NT9V5O9\"},\"namespace\":\"aws\"},{\"id\":
\"2279c0030c955e52\",\"name\":\"Initialization\",\"start_time\":1.499473412064E9,
\"end_time\":1.499473412819E9,\"aws\":{\"function_arn\":\"arn:aws:lambda:us-
west-2:123456789012:function:random-name\"}}]}",
                    "Id": "00f91aa01f4984fd"
                },
                {
                    "Document": "{\"id\":\"17ba309b32c7fbaf\",\"name\":
\"DynamoDB\",\"start_time\":1.49947341469E9,\"end_time\":1.499473414769E9,
\"parent_id\":\"4cd3f10b76c624b4\",\"inferred\":true,\"http\":{\"response
\":{\"status\":200,\"content_length\":57}},\"aws\":{\"table_name
\":\"scorekeep-user\",\"operation\":\"UpdateItem\",\"request_id\":
\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738BOB4KQNSO5AEMVJF66Q9\",\"resource_names\":
[\"scorekeep-user\"]},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"origin\":
\"AWS::DynamoDB::Table\"}",
                    "Id": "17ba309b32c7fbaf"
                },
                {
                    "Document": "{\"id\":\"1ee3c4a523f89ca5\",\"name\":\"SNS
\",\"start_time\":1.499473413112E9,\"end_time\":1.499473414071E9,\"parent_id\":
\"b29b548af4d54a0f\",\"inferred\":true,\"http\":{\"response\":{\"status\":200}},\"aws
\":{\"operation\":\"Publish\",\"region\":\"us-west-2\",\"request_id\":\"a2137970-
f6fc-5029-83e8-28aadeb99198\",\"retries\":0,\"topic_arn\":\"arn:aws:sns:us-
west-2:123456789012:awseb-e-ruag3jyweb-stack-NotificationTopic-6B829NT9V5O9\"},
\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"origin\":\"AWS::SNS\"}",
                    "Id": "1ee3c4a523f89ca5"
                }
            ],
            "Id": "1-59602603-23fc5b688855d396af79b496"
        }
    ],
    "UnprocessedTraceIds": []
}
```


Il tracciamento completo include un documento per ogni segmento, creato a partire da tutti i documenti di segmento con lo stesso ID di tracciamento ricevuti. Questi documenti non rappresentano i dati inviati a X-Ray dall'applicazione. Rappresentano invece i documenti elaborati

generati dal servizio X-Ray. X-Ray crea il documento di traccia completo compilando i documenti dei segmenti inviati dall'applicazione e rimuovendo i dati che non sono conformi allo schema del documento del segmento.

X-Ray crea anche segmenti dedotti per le chiamate downstream verso servizi che non inviano segmenti di per sé. Ad esempio, quando si chiama DynamoDB con un client strumentato, l'SDK X-Ray registra un sottosegmento con dettagli sulla chiamata dal suo punto di vista. Tuttavia, DynamoDB non invia un segmento corrispondente. X-Ray utilizza le informazioni nel sottosegmento per creare un segmento dedotto per rappresentare la risorsa DynamoDB nella mappa di traccia e lo aggiunge al documento di traccia.

Per ottenere più tracce dall'API, è necessario un elenco di tracce IDs, che è possibile estrarre dall'output di con una query. `get-trace-summaries`AWS CLI Reindirizza l'elenco all'input di `batch-get-traces` per ottenere tracce complete per un periodo di tempo specifico.

Example Script per ottenere i tracciamenti completi per un periodo di un minuto

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

Recupero e definizione dell’analisi delle cause principali

Dopo aver generato un riepilogo delle tracce con l'GetTraceSummaries API, è possibile riutilizzare i riepiloghi delle tracce parziali nel formato JSON per creare un'espressione di filtro raffinata basata sulle cause principali. Consulta gli esempi riportati di seguito per una procedura guidata per le fasi di definizione.

Example Esempio GetTraceSummaries di output: sezione relativa al tempo di risposta e alla causa principale

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

Modificando e apportando omissioni all'output precedente, questo JSON può diventare un filtro per le entità di cause principali corrispondenti. Per tutti i campi presenti nel JSON, ogni corrispondenza dei candidati deve essere esatta o il tracciamento non verrà restituito. I campi rimossi diventano valori jolly, un formato compatibile con la struttura di query dell'espressione di filtro.

Example Causa principale del tempo di risposta riformattata

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

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

Questo JSON viene quindi utilizzato come parte di un'espressione di filtro tramite una chiamata a `rootcause.json = #[{}]`. Consulta il capitolo Espressioni di filtro per ulteriori dettagli sulle query con espressioni di filtro.

Example Esempio di filtro JSON

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

Configurazione delle impostazioni di campionamento, gruppi e crittografia con l'API AWS X-Ray

AWS X-Ray consente di configurare APIs le regole di campionamento, le regole di gruppo e le impostazioni di crittografia.

Sections

- Impostazioni di crittografia
- Regole di campionamento
- Gruppi

Impostazioni di crittografia

Si usa `PutEncryptionConfig`per specificare una chiave AWS Key Management Service (AWS KMS) da usare per la crittografia.

> **Note**
>
> X-Ray non supporta le chiavi KMS asimmetriche.

```
$ 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"
    }
}
```

Come ID della chiave, puoi utilizzare un alias (come illustrato nell'esempio), una ID della chiave o di un ARN (Amazon Resource Name).

Utilizza `GetEncryptionConfig` per recuperare la configurazione corrente. Quando X-Ray termina di applicare le impostazioni, lo stato cambia da a`UPDATING`. `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"
    }
}
```

Per smettere di usare una chiave KMS e utilizzare la crittografia predefinita, imposta il tipo di crittografia su. `NONE`

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

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

Regole di campionamento

Puoi gestire le regole di campionamento nel tuo account con l'API X-Ray. Per ulteriori informazioni sull'aggiunta e la gestione dei tag, consulta. Etichettatura delle regole e dei gruppi di campionamento a raggi X

Recupera tutte le regole di campionamento utilizzando l'API `GetSamplingRules`.

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

La regola di default si applica a tutte le richieste che non soddisfano nessun'altra regola. Si tratta della regola a priorità più bassa e non può essere eliminata. Puoi, tuttavia, modificare la percentuale e la dimensione della riserva utilizzando `UpdateSamplingRule`.

Example Input API per: **`UpdateSamplingRule`**10000-default.json

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

L'esempio seguente utilizza il file precedente come input per modificare la regola di default impostando una percentuale dell'1% senza riserva. I tag sono opzionali. Se scegli di aggiungere tag, è necessaria una chiave di tag e i valori dei tag sono facoltativi. Per rimuovere i tag esistenti da una regola di campionamento, usa 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
        },
```

Crea regole di campionamento aggiuntive con `CreateSamplingRule`. Quando crei una regola, la maggior parte dei campi della regola sono obbligatori. L'esempio seguente crea due regole. Questa

prima regola imposta una percentuale fissa per l'applicazione di esempio Scorekeep. Si applica a tutte le richieste elaborate dall'API alle quali non si applicano regole di priorità maggiore.

Example Input API per: **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
    }
}
```

Anche la seconda regola si applica a Scorekeep, ma le è assegnata una priorità maggiore ed è più specifica. Questa regola consente di impostare una percentuale di campionamento molto bassa per le richieste di polling. Queste sono le richieste GET effettuate dal client ogni pochi secondi per verificare la presenza di modifiche allo stato del gioco.

Example Input API per **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
    }
```

```
}
```

I tag sono opzionali. Se scegli di aggiungere tag, è necessaria una chiave di tag e i valori dei tag sono facoltativi.

```
$ 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
    }
}
```

Per eliminare una regola di campionamento, utilizza `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
    }
}
```

Gruppi

Puoi utilizzare l'API X-Ray per gestire i gruppi nel tuo account. I gruppi costituiscono una raccolta di tracce definite da un'espressione di filtro. Puoi utilizzare i gruppi per generare grafici di servizio aggiuntivi e fornire CloudWatch metriche Amazon. Vedi Ottenere dati da AWS X-Ray per maggiori dettagli sull'utilizzo dei grafici e delle metriche dei servizi tramite l'API X-Ray. Per ulteriori informazioni

sui gruppi, vedere. Configurazione dei gruppi Per ulteriori informazioni sull'aggiunta e la gestione dei tag, vedereEtichettatura delle regole e dei gruppi di campionamento a raggi X.

Crea un gruppo con `CreateGroup`. I tag sono opzionali. Se scegli di aggiungere tag, è necessaria una chiave di tag e i valori dei tag sono facoltativi.

```
$ 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}"
}
```

Ottieni tutti i gruppi esistenti con `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"
}
```

Aggiorna un gruppo con `UpdateGroup`. I tag sono opzionali. Se scegli di aggiungere tag, è necessaria una chiave di tag e i valori dei tag sono facoltativi. Per rimuovere i tag esistenti da un gruppo, usa 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}"
}
```

Elimina un gruppo con `DeleteGroup`.

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

Using sampling rules with the X-Ray API (Utilizzo delle regole di campionamento con l'API X-Ray)

L' AWS X-Ray SDK utilizza l'API X-Ray per ottenere regole di campionamento, riportare i risultati del campionamento e ottenere quote. Puoi utilizzarli APIs per comprendere meglio come funzionano le regole di campionamento o per implementare il campionamento in un linguaggio non supportato da X-Ray SDK.

Inizia a recuperare tutte le regole di campionamento con `GetSamplingRules`.

```
$ aws xray get-sampling-rules
{
    "SamplingRuleRecords": [
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.01,
                "ReservoirSize": 0,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
```

```
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1530558121.0
        },
        {
            "SamplingRule": {
                "RuleName": "base-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/base-scorekeep",
                "ResourceARN": "*",
                "Priority": 9000,
                "FixedRate": 0.1,
                "ReservoirSize": 2,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530573954.0,
            "ModifiedAt": 1530920505.0
        },
        {
            "SamplingRule": {
                "RuleName": "polling-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/polling-scorekeep",
                "ResourceARN": "*",
                "Priority": 5000,
                "FixedRate": 0.003,
                "ReservoirSize": 0,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "GET",
                "URLPath": "/api/state/*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530918163.0,
            "ModifiedAt": 1530918163.0
        }
```

```
    ]
}
```

L'output include la regola di default e regole personalizzate. Se non hai ancora creato nessuna regola di campionamento, consulta Regole di campionamento.

Valuta le regole rispetto alle richieste in entrata in ordine crescente di priorità. Quando una regola viene soddisfatta, utilizza la percentuale fissa e la dimensione della riserva per prendere una decisione sul campionamento. Registra le richieste campionate e ignora (ai fini del tracciamento) le richieste non campionate. Interrompe la valutazione delle regole nel momento un cui viene assunta una decisione sul campionamento.

La dimensione della riserva di una regola è il numero di tracciamenti al secondo da registrare prima di applicare la percentuale fissa. La riserva è conteggiata su tutti i servizi in modo cumulativo, quindi non puoi utilizzarla direttamente. Tuttavia, se è diverso da zero, è possibile prendere in prestito una traccia al secondo dal serbatoio fino a quando X-Ray non assegna una quota. Prima di ricevere una quota, registra la prima richiesta ogni secondo e applica la percentuale fissa per ulteriori richieste. La percentuale fissa è un numero decimale tra 0 e 1,00 (100%).

L'esempio seguente mostra una chiamata a `GetSamplingTargets` con dettagli sulle decisioni di campionamento effettuate negli ultimi 10 secondi.

```
$ aws xray get-sampling-targets --sampling-statistics-documents '[
    {
        "RuleName": "base-scorekeep",
        "ClientID": "ABCDEF1234567890ABCDEF10",
        "Timestamp": "2018-07-07T00:20:06",
        "RequestCount": 110,
        "SampledCount": 20,
        "BorrowCount": 10
    },
    {
        "RuleName": "polling-scorekeep",
        "ClientID": "ABCDEF1234567890ABCDEF10",
        "Timestamp": "2018-07-07T00:20:06",
        "RequestCount": 10500,
        "SampledCount": 31,
        "BorrowCount": 0
    }
]'
```

```
{
    "SamplingTargetDocuments": [
        {
            "RuleName": "base-scorekeep",
            "FixedRate": 0.1,
            "ReservoirQuota": 2,
            "ReservoirQuotaTTL": 1530923107.0,
            "Interval": 10
        },
        {
            "RuleName": "polling-scorekeep",
            "FixedRate": 0.003,
            "ReservoirQuota": 0,
            "ReservoirQuotaTTL": 1530923107.0,
            "Interval": 10
        }
    ],
    "LastRuleModification": 1530920505.0,
    "UnprocessedStatistics": []
}
```

La risposta di X-Ray include una quota da utilizzare invece di prendere in prestito dal serbatoio. In questo esempio, il servizio ha prelevato 10 tracciamenti dalla riserva durante i 10 secondi, e viene applicata la percentuale fissa del 10% per le altre 100 richieste, generando un totale di 20 richieste campionate. La quota è valida per cinque minuti (indicata dal periodo di vita) o fino all'assegnazione di una nuova quota. X-Ray può anche assegnare un intervallo di segnalazione più lungo di quello predefinito, anche se qui non è così.

> **Note**
>
> La risposta di X-Ray potrebbe non includere una quota la prima volta che viene chiamata. Continua a prelevare dalla riserva finché non viene assegnata una quota.

Gli altri due campi della risposta possono indicare dei problemi con l'input. Controlla `LastRuleModification` rispetto all'ultima chiamata di `GetSamplingRules`. Se è più recente, scarica una nuova copia delle regole. `UnprocessedStatistics` può includere errori che indicano che una regola è stata eliminata, che il documento delle statistiche di input era troppo vecchio o che sono presenti dei problemi con le autorizzazioni.

AWS X-Ray segmentare i documenti

Un segmento di tracciamento è una rappresentazione JSON di una richiesta elaborata dalla tua applicazione. Un segmento di traccia registra informazioni sulla richiesta originale, informazioni sul lavoro svolto localmente dall'applicazione e sottosegmenti con informazioni sulle chiamate downstream effettuate dall'applicazione alle AWS risorse, ai database HTTP e SQL. APIs

Un documento di segmento trasmette informazioni su un segmento a X-Ray. Un documento di segmento può pesare fino a 64 kB e contenere un intero segmento con sottosegmenti, un frammento di segmento che indica che una richiesta è in corso o un singolo sottosegmento inviato separatamente. È possibile inviare documenti segmentati direttamente a X-Ray utilizzando l'`PutTraceSegments`API.

X-Ray compila ed elabora i documenti dei segmenti per generare riepiloghi delle tracce interrogabili e tracce complete a cui è possibile accedere utilizzando rispettivamente il comando e. `GetTraceSummaries` `BatchGetTraces` APIs Oltre ai segmenti e ai sottosegmenti inviati a X-Ray, il servizio utilizza le informazioni nei sottosegmenti per generare segmenti dedotti e li aggiunge alla traccia completa. I segmenti dedotti rappresentano i servizi e le risorse a valle nella mappa di traccia.

X-Ray fornisce uno schema JSON per i documenti dei segmenti. È possibile scaricare lo schema qui: xray-segmentdocument-schema-v 1.0.0. I campi e gli oggetti elencati nello schema sono descritti in modo dettagliato nelle sezioni seguenti.

Un sottoinsieme di campi di segmento viene indicizzato da X-Ray per essere utilizzato con le espressioni di filtro. Ad esempio, se imposti il `user` campo su un segmento su un identificatore univoco, puoi cercare segmenti associati a utenti specifici nella console X-Ray o utilizzando l'API. `GetTraceSummaries` Per ulteriori informazioni, consulta Utilizzo delle espressioni di filtro.

Quando strumentate la vostra applicazione con l'SDK X-Ray, l'SDK genera documenti di segmento per voi. Invece di inviare i documenti segmentati direttamente a X-Ray, l'SDK li trasmette tramite una porta UDP locale al demone X-Ray. Per ulteriori informazioni, consulta Invio di documenti di segmento al demone X-Ray.

Sections

- Campi del segmento
- Sottosegmenti
- Dati sulle richieste HTTP
- Annotazioni

- Metadati
- AWS dati relativi alle risorse
- Errori ed eccezioni
- Query SQL

Campi del segmento

Un segmento memorizza le informazioni di tracciamento relative a una richiesta elaborata dalla tua applicazione. Come minimo, un segmento memorizza nome, ID, ora di inizio, ID di tracciamento e ora di fine della richiesta.

Example Segmento completo minimale

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

Di seguito sono elencati i campi obbligatori, o condizionalmente obbligatori, per i segmenti.

> **Note**
>
> I valori devono essere stringhe (lunghe fino a un massimo di 250 caratteri) a meno che non sia indicato altrimenti.

Campi del segmento obbligatori

- `name`— Il nome logico del servizio che ha gestito la richiesta, fino a 200 caratteri. Ad esempio, il nome dell'applicazione o il nome di dominio. I nomi possono contenere lettere in formato Unicode, numeri e spazi e i seguenti simboli: `_`, `.`, `:`, `/`, `%`, `&`, `#`, `=`, `+`, `\`, `-`, `@`
- `id`— Un identificatore a 64 bit per il segmento, unico tra i segmenti della stessa traccia, in 16 cifre esadecimali.
- `trace_id`— Un identificatore univoco che collega tutti i segmenti e i sottosegmenti provenienti da una singola richiesta del client.

Formato ID di traccia X-Ray

Una radiografia `trace_id` è composta da tre numeri separati da trattini. Ad esempio `1-58406520-a006649127e371903a2de979`. Questo include:

- Il numero di versione, che è. 1
- L'ora della richiesta originale in Unix epoch time utilizzando 8 cifre esadecimali.

  Ad esempio, il fuso orario PST delle 10:00 del 1° dicembre 2016 è espresso in secondi o in cifre esadecimali. `1480615200` `58406520`
- Un identificatore a 96 bit univoco a livello globale per la traccia in 24 cifre esadecimali.

> **Note**
>
> X-Ray ora supporta IDs le tracce create utilizzando OpenTelemetry e qualsiasi altro framework conforme alla specifica W3C Trace Context. Un ID di traccia W3C deve essere formattato nel formato X-Ray Trace ID quando viene inviato a X-Ray. Ad esempio, l'ID di traccia W3C `4efaaf4d1e8720b39541901950019ee5` deve essere formattato come `1-4efaaf4d-1e8720b39541901950019ee5` quando si invia a X-Ray. La traccia X-Ray IDs include il timestamp originale della richiesta in Unix epoch Time, ma questo non è necessario quando si invia la traccia W3C in formato X-Ray. IDs

> **Sicurezza dell'ID di tracciamento**
>
> Le tracce sono visibili nelle intestazioni di risposta. IDs Genera tracce IDs con un algoritmo casuale sicuro per garantire che gli aggressori non possano calcolare le tracce future IDs e inviare richieste con quelle IDs alla tua applicazione.

- `start_time`— numero che indica l'ora in cui il segmento è stato creato, in secondi a virgola mobile nell'ora dell'epoca. Ad esempio `1480615200.010` o `1.480615200010E9`. Utilizza il numero di cifre decimali di cui hai bisogno. Quando disponibile, è consigliata la risoluzione al microsecondo.
- `end_time`— numero che indica l'ora in cui il segmento è stato chiuso. Ad esempio `1480615200.090` o `1.480615200090E9`. Specifica un `end_time` oppure il valore `in_progress`.

- `in_progress`— booleano, impostato su `true` invece di specificare an `end_time` per registrare che un segmento è iniziato, ma non è completo. Invia la segnalazione di un segmento in elaborazione per memorizzare la richiesta ricevuta quando l'applicazione riceve una richiesta la cui esecuzione richiede molto tempo. Quando viene inviata la risposta, invia il completamento del segmento per sovrascrivere il segmento in elaborazione. Per ogni richiesta, invia un solo segmento completo e uno o nessun segmento in esecuzione.

> **Nomi dei servizi**
>
> I segmenti `name` devono corrispondere al nome di dominio o al nome logico del servizio che genera il segmento. Tuttavia, questo non viene applicato. Qualsiasi applicazione autorizzata `PutTraceSegments`può inviare segmenti con qualsiasi nome.

I seguenti campi dei segmenti sono facoltativi.

Campi del segmento facoltativi

- `service`— Un oggetto con informazioni sull'applicazione.
  - `version`— Una stringa che identifica la versione dell'applicazione che ha fornito la richiesta.
- `user`— Una stringa che identifica l'utente che ha inviato la richiesta.
- `origin`— Il tipo di AWS risorsa che esegue l'applicazione.

  Valori supportati

  - `AWS::EC2::Instance`— Un' EC2 istanza Amazon.
  - `AWS::ECS::Container`— Un contenitore Amazon ECS.
  - `AWS::ElasticBeanstalk::Environment`— Un ambiente Elastic Beanstalk.

  Quando, per la tua applicazione, sono validi più valori, utilizza quello più specifico. Ad esempio, un ambiente Docker Elastic Beanstalk Multicontainer esegue l'applicazione su un contenitore Amazon ECS, che a sua volta viene eseguito su un'istanza Amazon. EC2 In questo caso, è possibile impostare l'origine a `AWS::ElasticBeanstalk::Environment` poiché l'ambiente è il padre delle altre due risorse.
- `parent_id`— Un ID di sottosegmento che specifichi se la richiesta proviene da un'applicazione strumentata. L'X-Ray SDK aggiunge l'ID del sottosegmento principale all'intestazione di

tracciamento per le chiamate HTTP downstream. Nel caso di sottosegmenti nidificati, un sottosegmento può avere come padre un segmento o un sottosegmento.

- `http`— `http`oggetti con informazioni sulla richiesta HTTP originale.
- `aws`— `aws`oggetto con informazioni sulla AWS risorsa su cui l'applicazione ha fornito la richiesta.
- `error`, `throttle`, `fault`, e `cause` — campi di errore che indicano che si è verificato un errore e che includono informazioni sull'eccezione che ha causato l'errore.
- `annotations`— `annotations`oggetto con coppie chiave-valore che si desidera che X-Ray indicizzi per la ricerca.
- `metadata`— `metadata`oggetto con eventuali dati aggiuntivi che si desidera memorizzare nel segmento.
- `subsegments`— matrice di `subsegment`oggetti.

Sottosegmenti

È possibile creare sottosegmenti per registrare le chiamate Servizi AWS e le risorse effettuate con l' AWS SDK, le chiamate al Web APIs HTTP interno o esterno o le query del database SQL. Puoi anche creare dei sottosegmenti per eseguire il debug o annotare blocchi di codice dell'applicazione. I sottosegmenti possono contenere a loro volta altri sottosegmenti, perciò un sottosegmento personalizzato che memorizza i metadati su una chiamata ad una funzione interna può contenere altri sottosegmenti personalizzati o sottosegmenti relativi alle chiamate a valle.

Un sottosegmento registra una chiamata a valle dal punto di vista del servizio che la chiama. X-Ray utilizza i sottosegmenti per identificare i servizi a valle che non inviano segmenti e creare le relative voci nel grafico dei servizi.

Un sottosegmento può essere incorporato in un intero documento di segmento oppure può essere inviato in modo indipendente. Invia sottosegmenti separatamente per tracciare in modo asincrono le chiamate a valle in caso di richieste di lunga durata, oppure per evitare di superare la dimensione massima del documento di segmento.

Example Segmento con sottosegmento incorporato

Un sottosegmento indipendente presenta un `type` valorizzato a `subsegment` e un `parent_id` che identifica il segmento padre.

```
{
```

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

Per le richieste di lunga durata, è possibile inviare un segmento in corso per notificare a X-Ray che la richiesta è stata ricevuta e quindi inviare i sottosegmenti separatamente per tracciarli prima di completare la richiesta originale.

Example Segmento in esecuzione

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

Example Sottosegmento indipendente

Un sottosegmento presenta un `type` valorizzato a `subsegment`, un `trace_id` e un `parent_id` che identifica il segmento padre.

```
{
  "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 la richiesta viene completata, chiudi il segmento inviandolo nuovamente con il campo `end_time` valorizzato. Il segmento completato sovrascrive il segmento in elaborazione.

Puoi anche inviare i sottosegmenti separatamente per richieste completate che hanno attivato flussi di lavoro asincroni. Ad esempio, un'API web potrebbe restituire immediatamente una risposta 0K

200 prima di avviare l'attività richiesta dall'utente. È possibile inviare un segmento completo a X-Ray non appena viene inviata la risposta, seguito da sottosegmenti per il lavoro completato in un secondo momento. Come nel caso dei segmenti, puoi anche inviare un sottosegmento per memorizzare l'inizio del sottosegmento stesso e quindi sovrascriverlo con un sottosegmento completo una volta completata la chiamata a valle.

Di seguito sono elencati i campi obbligatori, o condizionalmente obbligatori, per i sottosegmenti.

> **Note**
>
> I valori devono essere stringhe lunghe fino a un massimo di 250 caratteri, a meno che non sia indicato altrimenti.

Campi del sottosegmento obbligatori

- `id`— Un identificatore a 64 bit per il sottosegmento, unico tra i segmenti della stessa traccia, in 16 cifre esadecimali.
- `name`— Il nome logico del sottosegmento. Nel caso di chiamate a valle, denominare il sottosegmento in base alla risorsa o al servizio chiamati. Nel caso di sottosegmenti personalizzati, denominare il sottosegmento in base al codice che viene analizzato (ad esempio utilizzando un nome di funzione).
- `start_time`— numero che indica l'ora in cui è stato creato il sottosegmento, in secondi a virgola mobile in base all'ora dell'epoca, preciso in millisecondi. Ad esempio `1480615200.010` o `1.480615200010E9`.
- `end_time`— numero che indica l'ora in cui il sottosegmento è stato chiuso. Ad esempio `1480615200.090` o `1.480615200090E9`. Specificare un `end_time` o il valore `in_progress`.
- `in_progress`— valore booleano impostato su `true` anziché specificare an per `end_time` registrare che un sottosegmento è iniziato, ma non è completo. Per ogni richiesta a valle, invia un solo sottosegmento completo e uno o nessun sottosegmento in lavorazione.
- `trace_id`— ID di traccia del segmento principale del sottosegmento. Obbligatorio solo nel caso in cui si invii un sottosegmento in modo separato.

  Formato ID di traccia X-Ray

  Una radiografia `trace_id` è composta da tre numeri separati da trattini. Ad esempio `1-58406520-a006649127e371903a2de979`. Questo include:

- Il numero di versione, che è. 1
- L'ora della richiesta originale in Unix epoch time utilizzando 8 cifre esadecimali.

  Ad esempio, il fuso orario PST delle 10:00 del 1° dicembre 2016 è espresso in secondi o in cifre esadecimali. `1480615200` `58406520`
- Un identificatore a 96 bit univoco a livello globale per la traccia in 24 cifre esadecimali.

> **Note**
>
> X-Ray ora supporta IDs le tracce create utilizzando OpenTelemetry e qualsiasi altro framework conforme alla specifica W3C Trace Context. Un ID di traccia W3C deve essere formattato nel formato X-Ray Trace ID quando viene inviato a X-Ray. Ad esempio, l'ID di traccia W3C `4efaaf4d1e8720b39541901950019ee5` deve essere formattato come `1-4efaaf4d-1e8720b39541901950019ee5` quando si invia a X-Ray. La traccia X-Ray IDs include il timestamp originale della richiesta in Unix epoch Time, ma questo non è necessario quando si invia la traccia W3C in formato X-Ray. IDs

- `parent_id`— ID del segmento principale del sottosegmento. Obbligatorio solo nel caso in cui si invii un sottosegmento in modo separato. Nel caso di sottosegmenti nidificati, un sottosegmento può avere come padre un segmento o un sottosegmento.
- `type`—`subsegment`. Richiesto solo se si invia un sottosegmento separatamente.

I seguenti campi dei sottosegmenti sono facoltativi.

Campi del sottosegmento facoltativi

- `namespace`— `aws` per le chiamate SDK AWS; `remote` per altre chiamate downstream.
- `http`— `http`oggetto con informazioni su una chiamata HTTP in uscita.
- `aws`— `aws`oggetto con informazioni sulla AWS risorsa downstream chiamata dall'applicazione.
- `error`, `throttle`fault, e `cause` — campi di errore che indicano che si è verificato un errore e che includono informazioni sull'eccezione che ha causato l'errore.
- `annotations`— `annotations`oggetto con coppie chiave-valore che si desidera che X-Ray indicizzi per la ricerca.
- `metadata`— `metadata`oggetto con eventuali dati aggiuntivi che si desidera memorizzare nel segmento.
- `subsegments`— matrice di `subsegment`oggetti.

- `precursor_ids`— matrice di sottosegmenti IDs che identifica i sottosegmenti con lo stesso elemento principale completati prima di questo sottosegmento.

Dati sulle richieste HTTP

Utilizza un blocco HTTP per memorizzare i dettagli relativi a una richiesta HTTP che la tua applicazione ha elaborato (in un segmento) oppure che la tua applicazione ha eseguito a valle verso un'API HTTP (in un sottosegmento). La maggior parte dei campi di questo oggetto corrispondono alle informazioni di una richiesta HTTP e della relativa risposta.

`http`

Tutti i campi sono facoltativi.

- `request`— Informazioni su una richiesta.
  - `method`— Il metodo di richiesta. Ad esempio `GET`.
  - `url`— L'URL completo della richiesta, compilato a partire dal protocollo, dal nome host e dal percorso della richiesta.
  - `user_agent`— La stringa dell'agente utente proveniente dal client del richiedente.
  - `client_ip`— L'indirizzo IP del richiedente. Può essere recuperato dal campo `Source Address` del pacchetto IP o, in caso di richieste inoltrate, da un'intestazione `X-Forwarded-For`.
  - `x_forwarded_for`— (solo segmenti) booleano che indica che `client_ip` è stato letto da un'`X-Forwarded-For`intestazione e non è affidabile in quanto avrebbe potuto essere falsificato.
  - `traced`— (solo sottosegmenti) booleano che indica che la chiamata a valle è verso un altro servizio tracciato. Se questo campo è impostato su`true`, X-Ray considera la traccia interrotta finché il servizio downstream non carica un segmento con a `parent_id` che corrisponde al sottosegmento che `id` contiene questo blocco.
- `response`— Informazioni su una risposta.
  - `status`— numero intero che indica lo stato HTTP della risposta.
  - `content_length`— numero intero che indica la lunghezza del corpo della risposta in byte.

Quando effettui una chiamata a un'API Web a valle, registra un sottosegmento con informazioni sulla richiesta e sulla risposta HTTP. X-Ray utilizza il sottosegmento per generare un segmento dedotto per l'API remota.

Example Segmento per la chiamata HTTP servito da un'applicazione in esecuzione su 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 Sottosegmento per una chiamata HTTP a valle

```
{
  "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 dedotto per una chiamata HTTP a valle

```
{
  "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
}
```

Annotazioni

I segmenti e i sottosegmenti possono includere un `annotations` oggetto contenente uno o più campi indicizzati da X-Ray per l'uso con le espressioni di filtro. I campi possono avere valori di tipo stringa, numerico o booleano (senza oggetti o matrici). X-Ray indica fino a 50 annotazioni per traccia.

Example Segmento per chiamata HTTP con annotazioni

```
{
  "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
    }
  }
```

Per funzionare con i filtri, le chiavi devono essere di tipo alfanumerico. Il trattino basso è ammesso. Altri simboli e gli spazi non sono ammessi.

Metadati

I segmenti e i sottosegmenti possono includere un `metadata` oggetto contenente uno o più campi con valori di qualsiasi tipo, inclusi oggetti e matrici. X-Ray non indicizza i metadati e i valori possono avere qualsiasi dimensione, purché il documento del segmento non superi la dimensione massima (64 kB). Puoi visualizzare i metadati nel documento di segmento completo restituito dall'API

BatchGetTraces. Le chiavi di campo (debugnell'esempio seguente) che iniziano con AWS. sono riservate all'uso da AWS-provided e dai client SDKs .

Example Sottosegmento personalizzato con metadati

```
{
  "id": "0e58d2918e9038e8",
  "start_time": 1484789387.502,
  "end_time": 1484789387.534,
  "name": "## UserModel.saveUser",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
  "subsegments": [
    {
      "id": "0f910026178b71eb",
      "start_time": 1484789387.502,
      "end_time": 1484789387.534,
      "name": "DynamoDB",
      "namespace": "aws",
      "http": {
        "response": {
          "content_length": 58,
          "status": 200
        }
      },
      "aws": {
        "table_name": "scorekeep-user",
        "operation": "UpdateItem",
        "request_id": "3AIENM5J4ELQ3SPODHKBIRVIC3VV4KQNSO5AEMVJF66Q9ASUAAJG",
        "resource_names": [
          "scorekeep-user"
        ]
      }
    }
  ]
}
```

AWS dati relativi alle risorse

Nel caso dei segmenti, l'oggetto `aws` contiene informazioni sulla risorsa su cui è in esecuzione l'applicazione. A una singola risorsa possono essere associati molteplici campi. Ad esempio, un'applicazione in esecuzione in un ambiente Docker multicontainer su Elastic Beanstalk potrebbe contenere informazioni sull'istanza Amazon, sul contenitore Amazon ECS in esecuzione sull' EC2 istanza e sull'ambiente Elastic Beanstalk stesso.

`aws` (Segmenti)

Tutti i campi sono facoltativi.

- `account_id`— Se l'applicazione invia segmenti a un altro, registra l'ID dell'account su cui è in esecuzione l'applicazione. Account AWS
- `cloudwatch_logs`— Matrice di oggetti che descrivono un singolo gruppo di CloudWatch log.
  - `log_group`— Il nome del gruppo di CloudWatch log.
  - `arn`— L' CloudWatch ARN del gruppo Log.
- `ec2`— Informazioni su un' EC2 istanza Amazon.
  - `instance_id`— L'ID dell' EC2 istanza.
  - `instance_size`— Il tipo di EC2 istanza.
  - `ami_id`— L'Amazon Machine Image ID.
  - `availability_zone`— La zona di disponibilità in cui è in esecuzione l'istanza.
- `ecs`— Informazioni su un container Amazon ECS.
  - `container`— Il nome host del contenitore.
  - `container_id`— L'ID completo del contenitore.
  - `container_arn`— L'ARN dell'istanza del contenitore.
- `eks`— Informazioni su un cluster Amazon EKS.
  - pod— Il nome host del tuo pod EKS.
  - `cluster_name`— Il nome del cluster EKS.
  - `container_id`— L'ID completo del contenitore.
- `elastic_beanstalk`— Informazioni su un ambiente Elastic Beanstalk. Puoi trovare queste informazioni in un file denominato `/var/elasticbeanstalk/xray/environment.conf` sulle più recenti piattaforme Elastic Beanstalk.
  - `environment_name`: il nome dell'ambiente.

  - `version_label`— Il nome della versione dell'applicazione attualmente distribuita sull'istanza che ha fornito la richiesta.
  - `deployment_id`— numero che indica l'ID dell'ultima distribuzione riuscita nell'istanza che ha fornito la richiesta.
- `xray`— Metadati relativi al tipo e alla versione della strumentazione utilizzata.
  - `auto_instrumentation`— Booleano che indica se è stata utilizzata la strumentazione automatica (ad esempio, l'agente Java).
  - `sdk_version`— La versione dell'SDK o dell'agente in uso.
  - `sdk`— Il tipo di SDK.

Example AWS blocco con plugin

```
"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"
   }
}
```

Per i sottosegmenti, registra le informazioni sulle risorse Servizi AWS e sulle risorse a cui l'applicazione accede. X-Ray utilizza queste informazioni per creare segmenti dedotti che rappresentano i servizi a valle nella mappa dei servizi.

`aws` (Sottosegmenti)

Tutti i campi sono facoltativi.

- `operation`— Il nome dell'azione API richiamata contro una risorsa or. Servizio AWS
- `account_id`— Se l'applicazione accede alle risorse di un account diverso o invia segmenti a un account diverso, registra l'ID dell'account proprietario della AWS risorsa a cui l'applicazione ha avuto accesso.
- `region`— Se la risorsa si trova in una regione diversa dall'applicazione, registra la regione. Ad esempio `us-west-2`.
- `request_id`— Identificatore univoco per la richiesta.
- `queue_url`— Per le operazioni su una coda Amazon SQS, l'URL della coda.
- `table_name`— Per le operazioni su una tabella DynamoDB, il nome della tabella.

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento

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

Errori ed eccezioni

Quando si verifica un errore, puoi memorizzare le informazioni relative all'errore e le eccezioni generate. Memorizza gli errori nei segmenti quando l'applicazione restituisce un errore all'utente, e nei sottosegmenti quando l'errore è restituito da una chiamata a valle.

tipi di errore

Imposta uno o più dei seguenti campi a `true` per indicare che si è verificato un errore. In caso di errori composti è possibile associare molteplici tipi. Ad esempio, un errore `429 Too Many Requests` da una chiamata a valle può far sì che la tua applicazione ritorni un errore `500 Internal Server Error`, nel qual caso possono essere associati tutti e tre i tipi di errore.

- `error`— booleano che indica che si è verificato un errore del client (il codice di stato della risposta era 4XX Client Error).
- `throttle`— booleano che indica che una richiesta è stata limitata (il codice di stato della risposta era 429 Too Many Requests).
- `fault`— booleano che indica che si è verificato un errore del server (il codice di stato della risposta era 5XX Server Error).

Indica la causa dell'errore includendo un oggetto cause nel segmento o nel sottosegmento.

`cause`

A causa può essere un ID di eccezione lungo 16 caratteri o un oggetto con i campi riportati qui di seguito:

- `working_directory`— Il percorso completo della directory di lavoro quando si è verificata l'eccezione.
- `paths`— L'array di percorsi alle librerie o ai moduli in uso quando si è verificata l'eccezione.
- `exceptions`— La matrice di oggetti di eccezione.

Includi informazioni dettagliate sull'errore in uno o più oggetti exception.

`exception`

Tutti i campi sono facoltativi.

- `id`— Un identificatore a 64 bit per l'eccezione, unico tra i segmenti della stessa traccia, in 16 cifre esadecimali.
- `message`— Il messaggio di eccezione.
- `type`— Il tipo di eccezione.
- `remote`— booleano che indica che l'eccezione è stata causata da un errore restituito da un servizio a valle.
- `truncated`— numero intero che indica il numero di stack frame omessi da. `stack`
- `skipped`— numero intero che indica il numero di eccezioni che sono state saltate tra questa eccezione e la relativa eccezione figlia, ovvero l'eccezione che ha causato.
- `cause`— ID dell'eccezione principale dell'eccezione, ovvero l'eccezione che ha causato l'eccezione.
- `stack`— matrice di oggetti StackFrame.

Se disponibili, memorizza le informazioni sullo stack delle chiamate negli oggetti stackFrame.

`stackFrame`

Tutti i campi sono facoltativi.

- `path`— Il percorso relativo al file.
- `line`— La riga nel file.
- `label`— Il nome della funzione o del metodo.

Query SQL

Puoi creare o sottosegmenti per le query che la tua applicazione esegue su un database SQL.

`sql`

Tutti i campi sono facoltativi.

- `connection_string`— Per SQL Server o altre connessioni al database che non utilizzano stringhe di connessione URL, registrare la stringa di connessione, escluse le password.
- `url`— Per una connessione al database che utilizza una stringa di connessione URL, registrare l'URL, escluse le password.

- `sanitized_query`— L'interrogazione del database, con tutti i valori forniti dall'utente rimossi o sostituiti da un segnaposto.
- `database_type`— Il nome del motore del database.
- `database_version`— Il numero di versione del motore di database.
- `driver_version`— Il nome e il numero di versione del driver del motore di database utilizzato dall'applicazione.
- `user`— Il nome utente del database.
- `preparation`— `call` se la query utilizzava un`PreparedCall`; `statement` se l'interrogazione utilizzava un`PreparedStatement`.

Example Sottosegmento con una query SQL

```
{
  "id": "3fd8634e78ca9560",
  "start_time": 1484872218.696,
  "end_time": 1484872218.697,
  "name": "ebdb@aawijb5u25wdoy.cpamxznpdoq8.us-west-2.rds.amazonaws.com",
  "namespace": "remote",
  "sql" : {
    "url": "jdbc:postgresql://aawijb5u25wdoy.cpamxznpdoq8.us-
west-2.rds.amazonaws.com:5432/ebdb",
    "preparation": "statement",
    "database_type": "PostgreSQL",
    "database_version": "9.5.4",
    "driver_version": "PostgreSQL 9.4.1211.jre7",
    "user" : "dbuser",
    "sanitized_query" : "SELECT  *  FROM  customers  WHERE  customer_id=?;"
  }
}
```

AWS X-Ray concetti

AWS X-Ray riceve i dati dai servizi come segmenti. X-Ray quindi raggruppa i segmenti che hanno una richiesta comune in tracce. X-Ray elabora le tracce per generare un grafico del servizio che fornisce una rappresentazione visiva dell'applicazione.

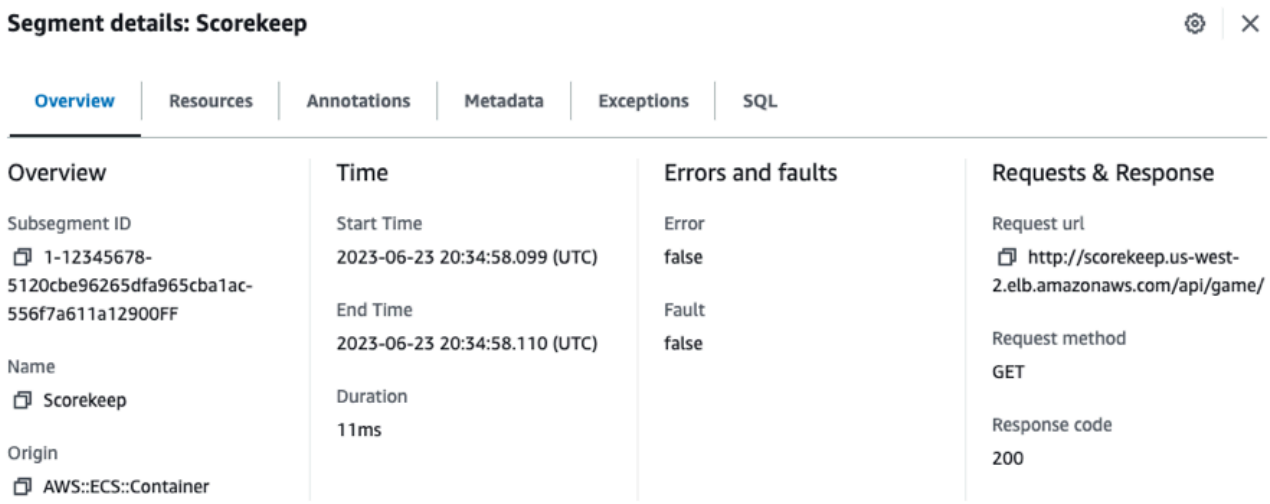
Concetti

- Segmenti
- Sottosegmenti
- Grafico del servizio
- Tracce
- Campionamento
- Intestazione di tracciamento
- Espressioni filtro
- Gruppi
- Annotazioni e metadati
- Errori, malfunzionamenti ed eccezioni

Segmenti

Le risorse di elaborazione che eseguono la logica dell'applicazione inviano i dati sulle loro attività sotto forma di segmenti. Un segmento contiene il nome della risorsa, i dettagli relativi alla richiesta e i dettagli sull'attività eseguita. Ad esempio, quando una richiesta HTTP raggiunge l'applicazione, il segmento può memorizzare i seguenti dati:

- L'host: nome host, alias o indirizzo IP
- La richiesta: metodo, indirizzo del cliente, percorso, agente utente
- La risposta: stato, contenuto
- Il lavoro svolto: orari di inizio e fine, sottosegmenti
- Problemi che si verificano: errori, guasti ed eccezioni, inclusa l'acquisizione automatica delle pile di eccezioni.



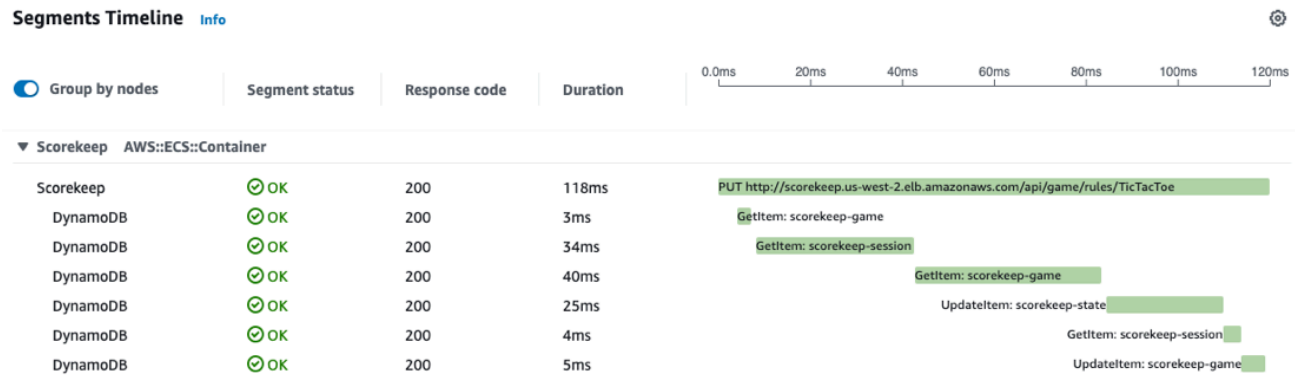
L' X-Ray SDK raccoglie informazioni dalle intestazioni di richiesta e risposta, dal codice dell'applicazione e dai metadati sulle AWS risorse su cui viene eseguito. Puoi scegliere i dati da raccogliere modificando la configurazione o il codice dell'applicazione per analizzare le richieste in arrivo, le richieste a valle e i client SDK. AWS

> **Richieste inoltrate**
>
> Se un load balancer o un altro intermediario inoltra una richiesta all'applicazione, X-Ray prende l'IP del client dall'`X-Forwarded-For`intestazione della richiesta anziché dall'IP di origine nel pacchetto IP. L'IP del client registrato per una richiesta inoltrata può essere contraffatto, quindi non dovrebbe essere considerato attendibile.

Puoi utilizzare l' X-Ray SDK per registrare informazioni aggiuntive come annotazioni e metadati. Per ulteriori informazioni sulla struttura e le informazioni memorizzate nei segmenti e nei sottosegmenti, consulta AWS X-Ray segmentare i documenti. I documenti di segmento possono avere una dimensione massima di 64 kB.

Sottosegmenti

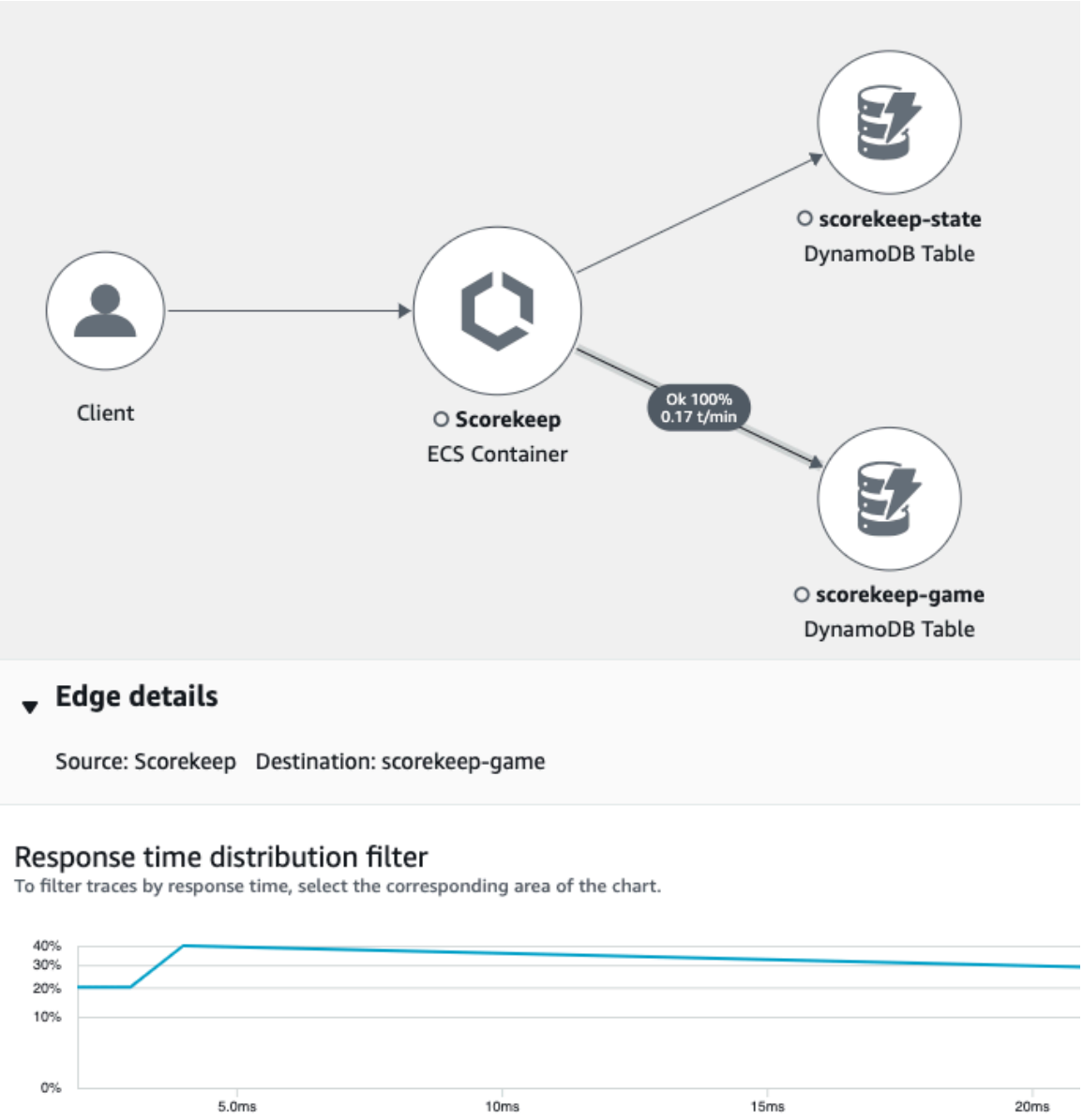
Un segmento può suddividere i dati sull'attività eseguita in sottosegmenti. I sottosegmenti offrono una maggiore granularità delle informazioni temporali e più dettagli sulle chiamate a valle invocate dall'applicazione per soddisfare la richiesta originale. Un sottosegmento può contenere dettagli aggiuntivi su una chiamata a un database SQL Servizio AWS, a un'API HTTP esterna o a un database SQL. Puoi anche definire sottosegmenti arbitrari per analizzare specifiche funzioni o righe di codice nell'applicazione.



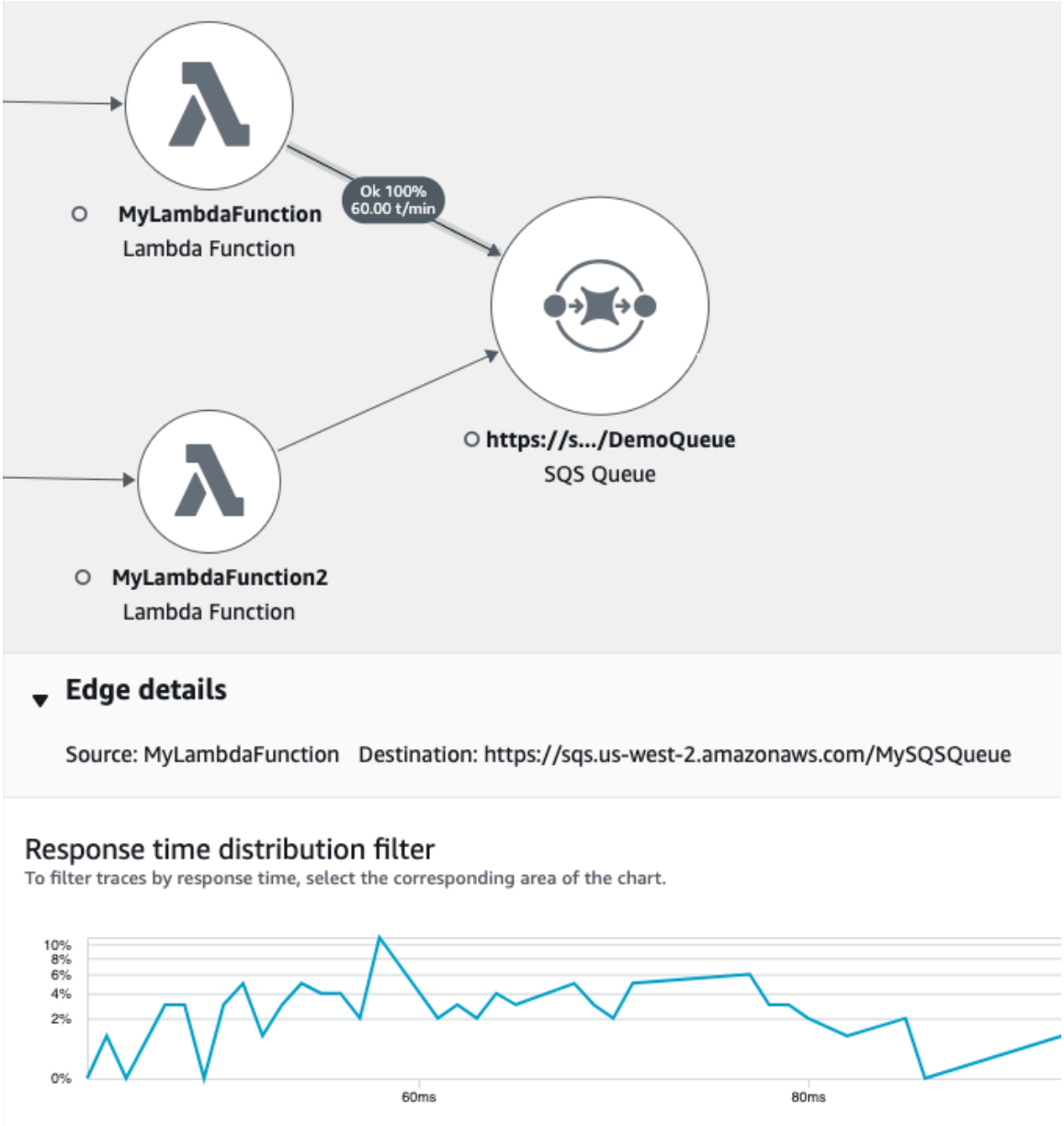
Per i servizi che non inviano segmenti propri, come Amazon DynamoDB, X-Ray utilizza i sottosegmenti per generare segmenti dedotti e nodi a valle sulla mappa di traccia. Questo ti consente di visualizzare tutte le dipendenze a valle, anche se non supportano il tracciamento o sono risorse esterne.

I sottosegmenti rappresentano la vista di una chiamata a valle da parte dell'applicazione che opera come client. Se anche il servizio a valle è analizzato, il segmento che questo invia sostituisce il segmento dedotto generato dal segmento del client a monte. Il nodo sul grafo del servizio utilizza sempre le informazioni del segmento del servizio, se disponibili, mentre il collegamento tra due nodi utilizza le informazioni del sottosegmento del servizio a monte.

Ad esempio, quando chiami DynamoDB con un client SDK strumentato, l' AWS SDK registra un sottosegmento per quella chiamata. X-Ray DynamoDB non invia un segmento, quindi il segmento dedotto nella traccia, il nodo DynamoDB nel grafico del servizio e il bordo tra il servizio e DynamoDB contengono tutti informazioni dal sottosegmento.



Quando chiami un altro servizio analizzato utilizzando con un'applicazione analizzata, il servizio a valle invia proprio segmento per memorizzare la sua vista della stessa chiamata che il servizio a monte ha memorizzato in un sottosegmento. Nel grafo del servizio, entrambi i nodi del servizio contengono le informazioni su tempi ed errori provenienti dai segmenti di tali servizi, mentre il collegamento tra essi è basato sulle informazioni provenienti dal sottosegmento del servizio a monte.



Entrambi i punti di vista sono utili, poiché il servizio a valle memorizza esattamente l'ora di inizio e fine dell'elaborazione della richiesta, mentre il servizio a monte memorizza la latenza totale che include il tempo che la richiesta ha consumato nel trasferimento tra i due servizi.

Grafico del servizio

X-Ray utilizza i dati inviati dall'applicazione per generare un grafico del servizio. Ogni AWS risorsa a cui invia i dati X-Ray viene visualizzata come servizio nel grafico. I collegamenti connettono i servizi che collaborano per elaborare le richieste. I collegamenti connettono i client alla tua applicazione e la tua applicazione ai servizi e alle risorse a valle utilizzate.

> **Nomi dei servizi**
>
> I segmenti `name` devono corrispondere al nome di dominio o al nome logico del servizio che genera il segmento. Tuttavia, questo non viene applicato. Qualsiasi applicazione autorizzata `PutTraceSegments` può inviare segmenti con qualsiasi nome.

Un grafo del servizio è un documento in formato JSON che contiene le informazioni sui servizi e sulle risorse che costituiscono la tua applicazione. La X-Ray console utilizza il grafico del servizio per generare una visualizzazione o una mappa dei servizi.



Per un'applicazione distribuita, X-Ray combina i nodi di tutti i servizi che elaborano le richieste con lo stesso ID di traccia in un unico grafico di servizio. Il primo servizio a cui accede la richiesta aggiunge un'intestazione di tracciamento che viene propagata tra il front-end e servizi che vengono invocati.

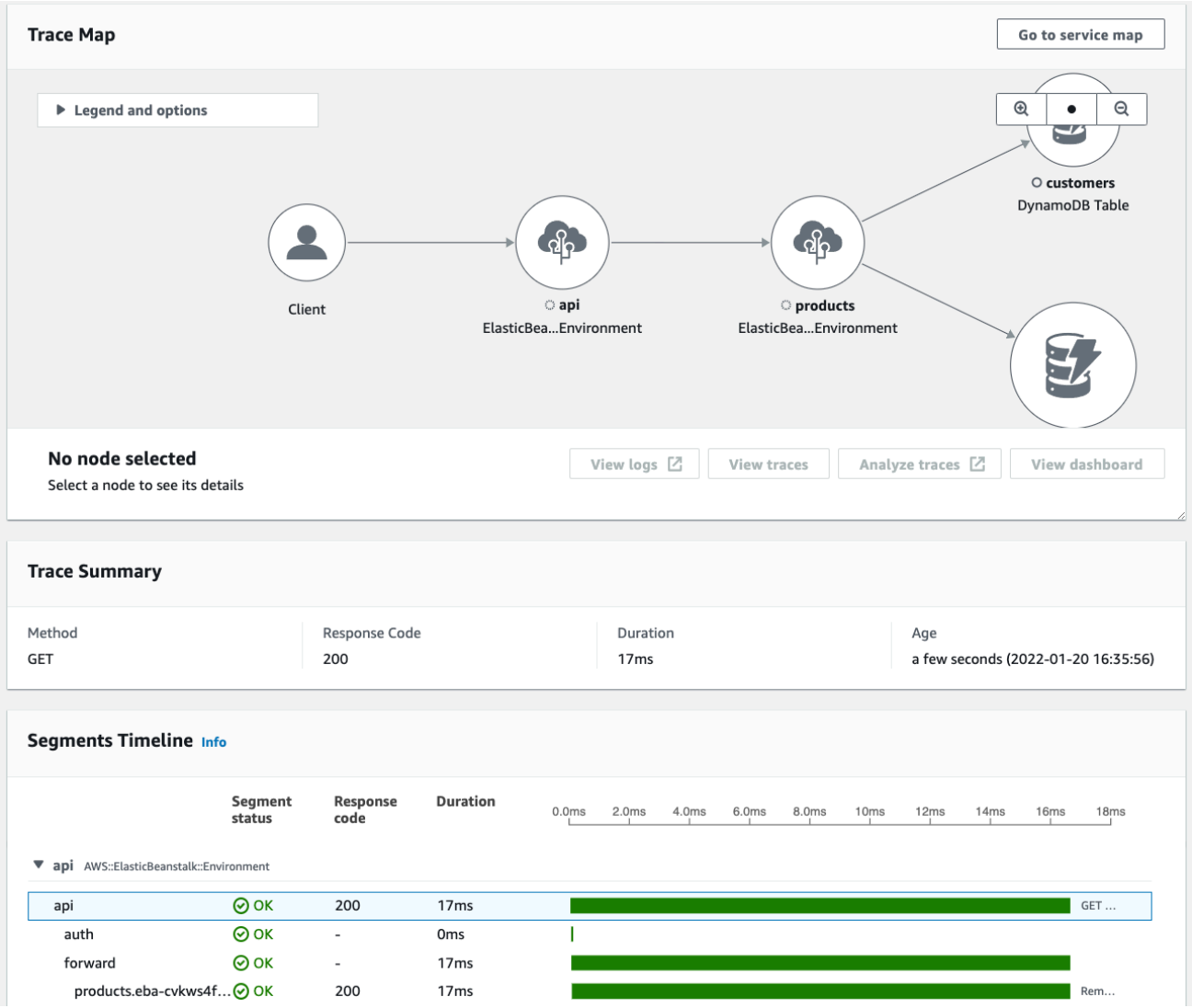
Ad esempio, Scorekeep esegue un'API Web che chiama un microservizio (una AWS Lambda funzione) per generare un nome casuale utilizzando una libreria. Node.js L' X-Ray SDK per Java genera l'ID di traccia e lo include nelle chiamate a Lambda. Lambda invia i dati di tracciamento e

passa l'ID di traccia alla funzione. L' X-Ray SDK per utilizza Node.js anche l'ID di traccia per inviare dati. Di conseguenza, i nodi per l'API, il servizio Lambda e la funzione Lambda appaiono tutti come nodi separati ma connessi sulla mappa di traccia.

I dati del grafico del servizio vengono conservati per 30 giorni.

Tracce

Un ID di tracciamento traccia il percorso di una richiesta attraverso l'applicazione. Un tracciamento raccoglie tutti i segmenti generati da una singola richiesta. Tale richiesta è in genere una richiesta HTTP GET o POST che passa attraverso un sistema di bilanciamento del carico, recupera il codice dell'applicazione e genera chiamate downstream ad altri AWS servizi o API Web esterne. Il primo servizio supportato con cui la richiesta HTTP interagisce aggiunge un'intestazione contenente un ID di tracciamento alla richiesta e la propaga a valle per tracciare latenza, completamento e altri dati relativi alla richiesta.



Consulta AWS X-Ray i prezzi per informazioni su come X-Ray vengono fatturate le tracce. I dati di tracciamento vengono conservati per 30 giorni.

Campionamento

Per garantire un tracciamento efficiente e fornire un campione rappresentativo delle richieste soddisfatte dall'applicazione, l' X-Ray SDK applica un algoritmo di campionamento per determinare quali richieste vengono tracciate. Per impostazione predefinita, l' X-Ray SDK registra la prima richiesta ogni secondo e il cinque percento di eventuali richieste aggiuntive.

Per evitare di incorrere in costi del servizio durante l'acquisizione delle nozioni di base, il tasso di campionamento di default è conservativo. È possibile configurare X-Ray per modificare la regola di

campionamento predefinita e configurare regole aggiuntive che applicano il campionamento in base alle proprietà del servizio o della richiesta.

Ad esempio, potresti voler disabilitare il campionamento e tracciare tutte le richieste di chiamate che modificano lo stato o gestiscono utenti o transazioni. In caso di elevati volumi di chiamate di sola lettura, come il polling in background, i controlli dello stato della connessione o la manutenzione della connessione, puoi campionare con una bassa percentuale e continuare comunque a disporre di un numero sufficiente di dati per rilevare eventuali problemi.

Per ulteriori informazioni, consulta Configurazione delle regole di campionamento di.

Intestazione di tracciamento

Tutte le richieste sono tracciate, fino a un valore minimo configurabile. Dopo aver raggiunto tale minimo, viene tracciata solo una percentuale delle richieste per evitare costi non necessari. La selezione per il campionamento e l'ID di tracciamento vengono aggiunti alle richieste HTTP nelle intestazioni di tracciamento denominate `X-Amzn-Trace-Id`. Il primo X-Ray-integrated servizio a cui viene inviata la richiesta aggiunge un'intestazione di tracciamento, che viene letta dall' X-Ray SDK e inclusa nella risposta.

Example Intestazione di tracciamento con ID di tracciamento root e selezione per il campionamento

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

> **Sicurezza dell'intestazione di tracciamento**
>
> Un'intestazione di tracciamento può provenire dall' X-Ray SDK, da una richiesta o dal client. Servizio AWS La tua applicazione può rimuovere l'intestazione `X-Amzn-Trace-Id` dalle richieste in entrata per evitare problemi causati dall'aggiunta alle richieste di ID di tracciamento e selezione per il campionamento da parte degli utenti.

Se la richiesta proviene da un'applicazione analizzata, l'intestazione di tracciamento può anche contenere un ID segmento padre. Ad esempio, se l'applicazione chiama un'API web HTTP downstream con un client HTTP strumentato, l' X-Ray SDK aggiunge l'ID del segmento per la richiesta originale all'intestazione di tracciamento della richiesta downstream. Un'applicazione

analizzata che elabora la richiesta a valle può memorizzare l'ID del segmento padre per collegare le due richieste.

Example Intestazione di tracciamento con ID di tracciamento root, ID del segmento padre e selezione per il campionamento

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

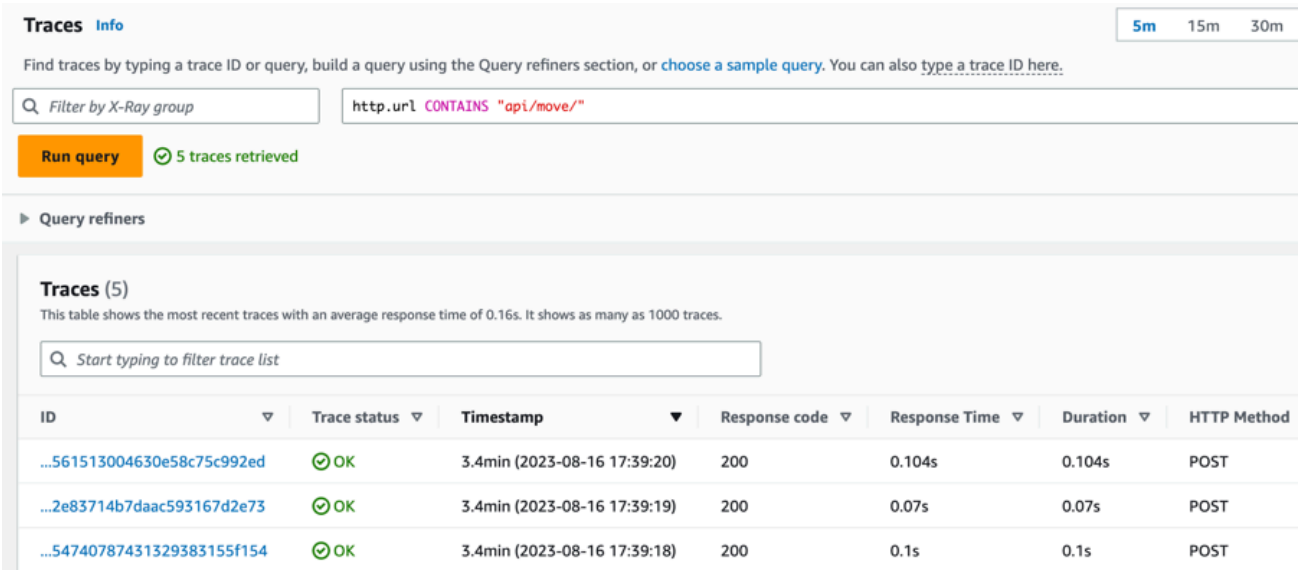
`Lineage`può essere aggiunto all'intestazione di traccia da Lambda e altri Servizi AWS come parte dei loro meccanismi di elaborazione e non deve essere utilizzato direttamente.

Example Intestazione di tracciamento con Lineage

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

Espressioni filtro

Anche utilizzando il campionamento, un'applicazione complessa genera una notevole quantità di dati. La AWS X-Ray console offre una visualizzazione facile da navigare del grafico del servizio. Mostra le informazioni di stato e prestazionali che ti permettono di identificare i problemi e le opportunità di ottimizzazione della tua applicazione. In caso di tracciamento avanzato, puoi approfondire l'analisi dei tracciamenti delle singole richieste, oppure utilizzare espressioni filtro per individuare tracciamenti correlati a specifici percorsi o utenti.



Gruppi

L'estensione delle espressioni di filtro supporta X-Ray anche la funzionalità di gruppo. Con un'espressione di filtro, è possibile definire i criteri in base ai quali accettare le tracce nel gruppo.

Puoi chiamare il gruppo per nome o tramite Amazon Resource Name (ARN) per generare il proprio grafico del servizio, i riepiloghi delle tracce e le metriche Amazon. CloudWatch Una volta creato un gruppo, le tracce in entrata vengono confrontate con l'espressione del filtro del gruppo mentre vengono archiviate nel servizio. X-Ray Le metriche relative al numero di tracce che corrispondono a ciascun criterio vengono pubblicate CloudWatch ogni minuto.

L'aggiornamento dell'espressione di filtro di un gruppo non modifica i dati già registrati. L'aggiornamento si applica solo per le successive tracce. Ciò può comportare un grafo unito delle espressioni nuove e vecchie. Per evitare questo problema, elimina il gruppo corrente e creane uno nuovo.

> **Note**
>
> I gruppi sono fatturati per il numero di tracce recuperate corrispondenti all'espressione di filtro. Per ulteriori informazioni, consulta Prezzi di AWS X-Ray.

Per ulteriori informazioni sui gruppi, vedereConfigurazione dei gruppi.

Annotazioni e metadati

Quando si esegue lo strumento dell'applicazione, l' X-Ray SDK registra le informazioni sulle richieste in entrata e in uscita, sulle AWS risorse utilizzate e sull'applicazione stessa. Puoi aggiungere al documento di segmento altre informazioni come annotazioni e metadati. Le annotazioni e i metadati vengono aggregati a livello di traccia e possono essere aggiunti a qualsiasi segmento o sottosegmento.

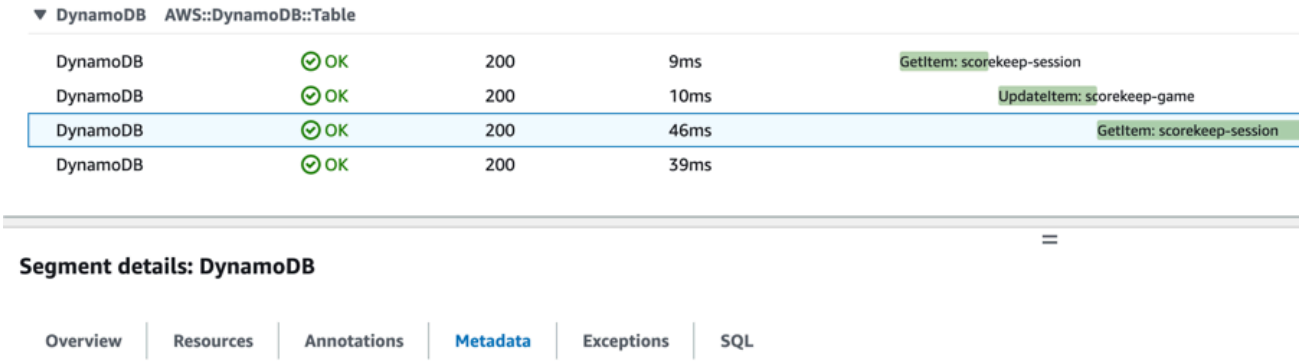
Le annotazioni sono semplici coppie chiave-valore indicizzate per l'uso con espressioni filtro. Utilizzale per registrare i dati che desideri utilizzare per raggruppare le tracce nella console oppure per chiamare l'API `GetTraceSummaries`.

X-Ray indicizza fino a 50 annotazioni per traccia.

I metadati sono coppie chiave-valore che possono assumere valori di qualsiasi tipo, inclusi oggetti ed elenchi, ma che non vengono indicizzate. Utilizza i metadati per registrare dati aggiuntivi che desideri

memorizzare all'interno del tracciamento ma non ti servono per effettuare ricerche all'interno dei tracciamenti.

È possibile visualizzare annotazioni e metadati nella finestra dei dettagli del segmento o del sottosegmento, all'interno della pagina dei dettagli della traccia nella console. CloudWatch



Errori, malfunzionamenti ed eccezioni

X-Ray tiene traccia degli errori che si verificano nel codice dell'applicazione e degli errori restituiti dai servizi downstream. Gli errori sono classificati come segue.

- **`Error`**— Errori del client (errori della serie 400)
- **`Fault`**— Guasti del server (errori della serie 500)
- **`Throttle`**— Errori di limitazione (429 richieste di troppo)

Quando si verifica un'eccezione mentre l'applicazione sta servendo una richiesta strumentata, l' X-Ray SDK registra i dettagli sull'eccezione, inclusa la traccia dello stack, se disponibile. Puoi visualizzare le eccezioni nei dettagli del segmento nella console. X-Ray

Sicurezza in AWS X-Ray

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gira Servizi AWS su Cloud AWS. AWS fornisce inoltre servizi che è possibile utilizzare in modo sicuro. L'efficacia della nostra sicurezza è regolarmente testata e verificata da revisori di terze parti come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità che si applicano a X-Ray, vedere Servizi AWS Scope by Compliance Program.
- Sicurezza nel cloud: la tua responsabilità è determinata da Servizio AWS ciò che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e leggi e normative applicabili.

Questa documentazione ti aiuterà a capire come applicare il modello di responsabilità condivisa quando usi X-Ray. I seguenti argomenti mostrano come configurare X-Ray per soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a utilizzarne altri Servizi AWS che possono aiutarti a monitorare e proteggere le tue risorse X-Ray.

Argomenti

- Protezione dei dati in AWS X-Ray
- Gestione delle identità e degli accessi per AWS X-Ray
- Convalida della conformità per AWS X-Ray
- Resilienza in AWS X-Ray
- Sicurezza dell'infrastruttura in AWS X-Ray

Protezione dei dati in AWS X-Ray

AWS X-Ray crittografa sempre le tracce e i dati correlati quando sono inattivi. Quando è necessario controllare e disabilitare le chiavi di crittografia per motivi di conformità o requisiti interni, è possibile

configurare X-Ray per utilizzare una chiave AWS Key Management Service (AWS KMS) per crittografare i dati.

X-Ray fornisce un Chiave gestita da AWS nome. `aws/xray` Utilizza questa chiave se desideri controllare l'utilizzo delle chiavi in AWS CloudTrail e non hai bisogno di gestire la chiave stessa. Quando è necessario gestire l'accesso alla chiave o configurare la rotazione delle chiavi, è possibile creare una chiave gestita dal cliente.

Quando si modificano le impostazioni di crittografia, X-Ray impiega un po' di tempo a generare e propagare le chiavi di dati. Anche se la nuova chiave è in corso di elaborazione, X-Ray può crittografare i dati con una combinazione di nuove e vecchie impostazioni. I dati esistenti non vengono crittografati nuovamente quando si modificano le impostazioni crittografiche.

> **Note**
>
> AWS KMS addebita quando X-Ray utilizza una chiave KMS per crittografare o decrittografare i dati di traccia.
>
> - Crittografia predefinita: gratuita.
> - Chiave gestita da AWS— Paga per l'uso della chiave.
> - chiave gestita dal cliente: paga per l'archiviazione e l'utilizzo delle chiavi.
>
> Per maggiori dettagli, consulta la sezione AWS Key Management Service Prezzi.

> **Note**
>
> Le notifiche X-Ray Insights inviano eventi ad Amazon EventBridge, che attualmente non supporta le chiavi gestite dai clienti. Per ulteriori informazioni, consulta la sezione Protezione dei dati in Amazon EventBridge.

È necessario disporre dell'accesso a livello utente a una chiave gestita dal cliente per configurare X-Ray per utilizzarla e quindi visualizzare le tracce crittografate. Per ulteriori informazioni, consulta Autorizzazioni utente per la crittografia.

CloudWatch console

Per configurare X-Ray per l'utilizzo di una chiave KMS per la crittografia tramite la console CloudWatch

1. Accedi a Console di gestione AWS e apri la CloudWatch console all'indirizzo. https://console.aws.amazon.com/cloudwatch/
2. Scegli Impostazioni nel riquadro di navigazione a sinistra.
3. Scegli Visualizza impostazioni in Crittografia nella sezione Tracce X-Ray.
4. Scegli Modifica nella sezione Configurazione della crittografia.
5. Scegli Usa una chiave KMS.
6. Scegliere una chiave dal menu a discesa:
   - aws/xray — Usa il. Chiave gestita da AWS
   - alias chiave: utilizza una chiave gestita dal cliente nel tuo account.
   - Inserisci manualmente una chiave ARN: utilizza una chiave gestita dal cliente in un altro account. Inserire l'ARN (Amazon Resource Name) completo della chiave nel campo visualizzato.
7. Scegli Aggiorna crittografia.

X-Ray console

Per configurare X-Ray per l'utilizzo di una chiave KMS per la crittografia utilizzando la console X-Ray

1. Apri la console X-Ray.
2. Scegliere Encryption (Crittografia).
3. Scegli Usa una chiave KMS.
4. Scegliere una chiave dal menu a discesa:
   - aws/xray — Usa il. Chiave gestita da AWS
   - alias chiave: utilizza una chiave gestita dal cliente nel tuo account.
   - Inserisci manualmente una chiave ARN: utilizza una chiave gestita dal cliente in un altro account. Inserire l'ARN (Amazon Resource Name) completo della chiave nel campo visualizzato.

5. Scegli Applica.

> **Note**
>
> X-Ray non supporta le chiavi KMS asimmetriche.

Se X-Ray non è in grado di accedere alla chiave di crittografia, interrompe l'archiviazione dei dati. Ciò può accadere se l'utente perde l'accesso alla chiave KMS o se disabiliti una chiave attualmente in uso. Quando ciò accade, X-Ray mostra una notifica nella barra di navigazione.

Per configurare le impostazioni di crittografia con l'API X-Ray, vedere. Configurazione delle impostazioni di campionamento, gruppi e crittografia con l'API AWS X-Ray

Gestione delle identità e degli accessi per AWS X-Ray

AWS Identity and Access Management (IAM) è uno strumento Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (dispone delle autorizzazioni) a utilizzare le risorse X-Ray. IAM è uno strumento Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell'accesso tramite policy
- Come AWS X-Ray funziona con IAM
- AWS X-Ray esempi di politiche basate sull'identità
- Risoluzione dei problemi di AWS X-Ray identità e accesso

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi di AWS X-Ray identità e accesso)

- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come AWS X-Ray funziona con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta AWS X-Ray esempi di politiche basate sull'identità)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come Centro identità AWS IAM (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Liste di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come AWS X-Ray funziona con IAM

Prima di utilizzare IAM per gestire l'accesso a X-Ray, è necessario comprendere quali funzionalità IAM sono disponibili per l'uso con X-Ray. Per avere una visione di alto livello del funzionamento di X-Ray e di Servizi AWS altro tipo con IAM, Servizi AWS consulta That Work with IAM nella IAM User Guide.

Puoi usare AWS Identity and Access Management (IAM) per concedere le autorizzazioni X-Ray agli utenti e alle risorse di calcolo del tuo account. IAM controlla l'accesso al servizio X-Ray a livello di API per applicare le autorizzazioni in modo uniforme, indipendentemente dal client (console, AWS SDK) impiegato dagli utenti. AWS CLI

Per utilizzare la console X-Ray per visualizzare mappe e segmenti di tracciamento, sono necessarie solo le autorizzazioni di lettura. Per abilitare l'accesso alla console, aggiungi la policy `AWSXrayReadOnlyAccess` gestita al tuo utente IAM.

Per lo sviluppo e il test locali, crea un ruolo IAM con autorizzazioni di lettura e scrittura. Assumi il ruolo e archivia le credenziali temporanee per il ruolo. È possibile utilizzare queste credenziali con il demone X-Ray, con e con AWS CLI l'SDK. AWS Per ulteriori informazioni, vedere Utilizzo di credenziali di sicurezza temporanee con. AWS CLI

Per distribuire la tua app strumentata su AWS, crea un ruolo IAM con autorizzazioni di scrittura e assegnalo alle risorse che eseguono l'applicazione. `AWSXRayDaemonWriteAccess`include l'autorizzazione a caricare tracce e alcune autorizzazioni di lettura per supportare l'uso delle regole di campionamento.

Le policy in lettura e in scrittura non includono l'autorizzazione per configurare le impostazioni delle chiavi di cifratura e le regole di campionatura. Utilizzalo `AWSXrayFullAccess` per accedere a queste impostazioni o aggiungere la configurazione APIs in una politica personalizzata. Per la crittografia e la decodifica con una chiave gestita personalizzata che crei, hai anche bisogno dell'autorizzazione per l'utilizzo della chiave.

Argomenti

- Policy basate sull'identità X-Ray

- Policy basate sulle risorse X-Ray
- Autorizzazione basata su tag X-Ray
- Esecuzione locale dell'applicazione
- Esecuzione dell'applicazione in AWS
- Autorizzazioni utente per la crittografia

Policy basate sull'identità X-Ray

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. X-Ray supporta azioni, risorse e chiavi di condizione specifiche. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente IAM.

Azioni

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Le azioni politiche in X-Ray utilizzano il seguente prefisso prima dell'azione:. `xray:` Ad esempio, per concedere a qualcuno l'autorizzazione a recuperare i dettagli delle risorse di gruppo con l'operazione dell'API `GetGroup` X-Ray, includi `xray:GetGroup` l'azione nella sua politica. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`. X-Ray definisce il proprio set di azioni che descrivono le attività che è possibile eseguire con questo servizio.

Per specificare più azioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

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

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Get`, includi la seguente azione:

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

Per visualizzare un elenco di azioni X-Ray, consulta Actions Defined by AWS X-Ray nella IAM User Guide.

Resources

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

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

Puoi controllare l'accesso alle risorse utilizzando una policy IAM. Per le operazioni che supportano le autorizzazioni a livello di risorsa, si utilizza un Amazon Resource Name (ARN) per identificare la risorsa a cui si applica la policy.

Tutte le azioni X-Ray possono essere utilizzate in una policy IAM per concedere o negare agli utenti il permesso di utilizzare quell'azione. Tuttavia, non tutte le azioni X-Ray supportano le autorizzazioni a livello di risorsa, che consentono di specificare le risorse su cui è possibile eseguire un'azione.

Per le operazioni che non supportano le autorizzazioni a livello di risorsa, è necessario utilizzare `"*"` come risorsa.

Le seguenti azioni X-Ray supportano le autorizzazioni a livello di risorsa:

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

Di seguito è riportato un esempio di policy di autorizzazione basata su identità per un'operazione `CreateGroup`. L'esempio mostra l'uso di un ARN relativo al nome di gruppo `local-users` con l'ID univoco come carattere jolly. L'ID univoco viene generato quando il gruppo viene creato, quindi non può essere previsto nella policy in anticipo. Quando utilizzi `GetGroup`, `UpdateGroup` oppure `DeleteGroup`, è possibile definirlo come un carattere jolly o l'ARN esatto, incluso l'ID.

> **Note**
>
> L'ARN di una regola di campionamento è definito in base al nome. A differenza del gruppo ARNs, le regole di campionamento non hanno un ID generato in modo univoco.

Per visualizzare un elenco dei tipi di risorse X-Ray e relativi ARNs, consulta Resources Defined by AWS X-Ray nella IAM User Guide. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consulta Operazioni definite da AWS X-Ray.

Chiavi di condizione

X-Ray non fornisce chiavi di condizione specifiche del servizio, ma supporta l'utilizzo di alcune chiavi di condizione globali. Per visualizzare tutte le chiavi di condizione AWS globali, consulta AWS Global Condition Context Keys nella IAM User Guide.

Esempi

Per visualizzare esempi di policy basate sull'identità a raggi X, vedere. AWS X-Ray esempi di politiche basate sull'identità

Policy basate sulle risorse X-Ray

X-Ray supporta politiche basate sulle risorse per l' Servizio AWS integrazione attuale e futura, come il tracciamento attivo di Amazon SNS. Le policy basate sulle risorse X-Ray possono essere aggiornate da altri Console di gestione AWS utenti o tramite l'SDK o la CLI. AWS Ad esempio, la console Amazon SNS tenta di configurare automaticamente una policy basata sulle risorse per l'invio di tracce a X-Ray. Il seguente documento sulla policy fornisce un esempio di configurazione manuale delle policy basate sulle risorse X-Ray.

Example Esempio di policy basata su risorse X-Ray per il tracciamento attivo di Amazon SNS

Questo documento di policy di esempio specifica le autorizzazioni necessarie ad Amazon SNS per inviare dati di traccia a X-Ray:

```
{
    Version: "2012-10-17",
    Statement: [
      {
        Sid: "SNSAccess",
        Effect: Allow,
        Principal: {
          Service: "sns.amazonaws.com",
        },
        Action: [
          "xray:PutTraceSegments",
          "xray:GetSamplingRules",
          "xray:GetSamplingTargets"
        ],
        Resource: "*",
        Condition: {
          StringEquals: {
            "aws:SourceAccount": "account-id"
          },
          StringLike: {
            "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
          }
        }
      }
    ]
  }
```

Utilizza la CLI per creare una policy basata sulle risorse che conceda ad Amazon SNS le autorizzazioni per inviare dati di traccia a X-Ray:

```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
 '{ "Version": "2012-10-17",         "Statement": [ { "Sid": "SNSAccess",
 "Effect": "Allow", "Principal": { "Service": "sns.amazonaws.com" }, "Action":
 [ "xray:PutTraceSegments", "xray:GetSamplingRules", "xray:GetSamplingTargets" ],
 "Resource": "*", "Condition": { "StringEquals": { "aws:SourceAccount": "account-
id" }, "StringLike": { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-
name" } } } ] }'
```

Per utilizzare questi esempi, sostituisci*partition*, *regionaccount-id*, e *topic-name* con la AWS partizione, la regione, l'ID dell'account e il nome dell'argomento Amazon SNS specifici. Per autorizzare tutti gli argomenti di Amazon SNS a inviare dati di traccia a X-Ray, sostituisci il nome dell'argomento con. *

Autorizzazione basata su tag X-Ray

È possibile allegare tag ai gruppi di raggi X o alle regole di campionamento oppure passare i tag in una richiesta a X-Ray. Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `xray:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Per ulteriori informazioni sull'etichettatura delle risorse X-Ray, vedere. Etichettatura delle regole e dei gruppi di campionamento a raggi X

Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Gestione dell'accesso ai gruppi X-Ray e alle regole di campionamento basate sui tag.

Esecuzione locale dell'applicazione

L'applicazione strumentata invia i dati di traccia al demone X-Ray. Il daemon memorizza i documenti segmentati e li carica sul servizio X-Ray in batch. Il demone necessita di autorizzazioni di scrittura per caricare dati di traccia e telemetria sul servizio X-Ray.

Quando esegui il daemon localmente, crea un ruolo IAM, assumi il ruolo e archivia le credenziali temporanee nelle variabili di ambiente o in un file denominato all'interno di una cartella denominata nella cartella utente. `credentials .aws` Per ulteriori informazioni, consulta Utilizzo di credenziali di sicurezza temporanee con. AWS CLI

Example~/.aws/credentials

```
[default]
aws_access_key_id={access key ID}
aws_secret_access_key={access key}
aws_session_token={AWS session token}
```

Se hai già configurato le credenziali da utilizzare con l' AWS SDK oppure AWS CLI il demone può utilizzarle. Se sono disponibili più profili, il daemon usa il profilo di default.

Esecuzione dell'applicazione in AWS

Quando esegui l'applicazione AWS, usa un ruolo per concedere l'autorizzazione all'istanza Amazon EC2 o alla funzione Lambda che esegue il demone.

- Amazon Elastic Compute Cloud (Amazon EC2): crea un ruolo IAM e collegalo all'istanza EC2 come profilo dell'istanza.

- Amazon Elastic Container Service (Amazon ECS): crea un ruolo IAM e collegalo alle istanze del contenitore come ruolo IAM dell'istanza di contenitore.
- AWS Elastic Beanstalk (Elastic Beanstalk) — Elastic Beanstalk include le autorizzazioni X-Ray nel profilo di istanza predefinito. Puoi utilizzare l'impostazione predefinita del profilo dell'istanza o aggiungere autorizzazioni di scrittura a un profilo di istanze personalizzato.
- AWS Lambda (Lambda) — Aggiungi i permessi di scrittura al ruolo di esecuzione della funzione.

Per creare un ruolo da utilizzare con X-Ray

1. Apri la console IAM.
2. Scegli Ruoli.
3. Scegli Crea nuovo ruolo.
4. Per Role Name (Nome ruolo) digitare **`xray-application`**. Selezionare Next Step (Fase successiva).
5. Come Role Type (Tipo ruolo), scegliere Amazon EC2.
6. Allega la seguente policy gestita per consentire all'applicazione di accedere a Servizi AWS:

   - AWSXRayDaemonWriteAccess— Fornisce al demone X-Ray il permesso di caricare dati di traccia.

   Se l'applicazione utilizza l' AWS SDK per accedere ad altri servizi, aggiungete politiche che consentano l'accesso a tali servizi.
7. Selezionare Next Step (Fase successiva).
8. Selezionare Crea ruolo.

Autorizzazioni utente per la crittografia

X-Ray crittografa tutti i dati di traccia e, per impostazione predefinita, è possibile configurarlo per utilizzare una chiave gestita dall'utente. Se scegli una chiave gestita AWS Key Management Service dal cliente, devi assicurarti che la politica di accesso della chiave consenta di concedere l'autorizzazione a X-Ray per utilizzarla per la crittografia. Anche gli altri utenti del tuo account devono accedere alla chiave per visualizzare i dati di traccia crittografati nella console X-Ray.

Per una chiave gestita dal cliente, configura la chiave con una politica di accesso che consenta le seguenti azioni:

- L'utente che configura la chiave in X-Ray è autorizzato a `kms:CreateGrant` chiamare e. `kms:DescribeKey`
- Gli utenti che accedono ai dati di tracciamento crittografati hanno l'autorizzazione di chiamare `kms:Decrypt`.

Quando aggiungi un utente al gruppo Key users nella sezione di configurazione delle chiavi della console IAM, dispone dell'autorizzazione per entrambe queste operazioni. L'autorizzazione deve essere impostata solo sulla policy chiave, quindi non è necessaria alcuna AWS KMS autorizzazione per utenti, gruppi o ruoli. Per ulteriori informazioni, consulta Using Key Policies nella AWS KMS Developer Guide.

Per la crittografia predefinita, o se si sceglie la CMK AWS gestita (`aws/xray`), l'autorizzazione si basa su chi ha accesso a X-Ray APIs. Tutti gli utenti con accesso a `PutEncryptionConfig`, inclusi in `AWSXrayFullAccess`, possono modificare la configurazione della crittografia. Per impedire a un utente di modificare la chiave crittografica, non assegnargli l'autorizzazione ad utilizzare `PutEncryptionConfig`.

AWS X-Ray esempi di politiche basate sull'identità

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse X-Ray. Inoltre, non possono eseguire attività utilizzando l' AWS API Console di gestione AWS AWS CLI, o. Un amministratore deve creare le policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore devi quindi collegare queste policy a utenti o gruppi che richiedono tali autorizzazioni.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy nella scheda JSON nella Guida per l'utente IAM.

Argomenti

- Best practice delle policy
- Uso della console X-Ray
- Consentire agli utenti di visualizzare le loro autorizzazioni
- Gestione dell'accesso ai gruppi X-Ray e alle regole di campionamento basate sui tag
- Policy gestite da IAM per X-Ray

- Aggiornamenti X-Ray alle AWS policy gestite
- Specificare una risorsa all'interno di una policy IAM

Best practice delle policy

Le policy basate sull'identità determinano se qualcuno può creare, accedere o eliminare le risorse X-Ray nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando

vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Uso della console X-Ray

Per accedere alla AWS X-Ray console, è necessario disporre di un set minimo di autorizzazioni. Queste autorizzazioni devono consentire all'utente di elencare e visualizzare i dettagli sulle risorse X-Ray presenti nel computer. Account AWS Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Per garantire che tali entità possano ancora utilizzare la console X-Ray, allega la policy `AWSXRayReadOnlyAccess` AWS gestita alle entità. Questa policy è descritta più dettagliatamente nelle policy gestite da IAM per X-Ray. Per ulteriori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente IAM.

Non è necessario consentire le autorizzazioni minime della console per gli utenti che effettuano chiamate solo verso AWS CLI o l' AWS API. Al contrario, è possibile accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI 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": "*"
        }
    ]
}
```

Gestione dell'accesso ai gruppi X-Ray e alle regole di campionamento basate sui tag

È possibile utilizzare le condizioni della policy basata sull'identità per controllare l'accesso ai gruppi X-Ray e le regole di campionamento basate sui tag. La seguente politica di esempio potrebbe essere utilizzata per negare a un ruolo utente le autorizzazioni per creare, eliminare o aggiornare gruppi con i tag o. `stage:prod` `stage:preprod` Per ulteriori informazioni sull'etichettatura delle regole e dei gruppi di campionamento a raggi X, vedere. Etichettatura delle regole e dei gruppi di campionamento a raggi X

Per negare la creazione di una regola di campionamento, utilizzate `aws:RequestTag` per indicare i tag che non possono essere passati come parte di una richiesta di creazione. Per negare l'aggiornamento o l'eliminazione di una regola di campionamento, utilizza l'opzione `aws:ResourceTag` per negare le azioni basate sui tag presenti su tali risorse.

Puoi allegare queste politiche (o combinarle in un'unica politica, quindi allegare la politica) agli utenti del tuo account. Affinché l'utente possa apportare modifiche a un gruppo o a una regola di campionamento, il gruppo o la regola di campionamento non devono essere etichettati `stage=prepod` o. `stage=prod` La chiave di tag di condizione `Stage` corrisponde a `Stage` e `stage`

perché i nomi delle chiavi di condizione non effettuano la distinzione tra maiuscole e minuscole. Per ulteriori informazioni sul blocco condition, consulta IAM JSON Policy Elements: Condition nella IAM User Guide.

Un utente con un ruolo a cui è associata la seguente policy non può aggiungere il tag `role:admin` alle risorse e non può rimuovere i tag da una risorsa a cui è `role:admin` associato.

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"
                }
            }
        }
    ]
}
```

Policy gestite da IAM per X-Ray

Per semplificare la concessione delle autorizzazioni, IAM supporta le policy gestite per ogni servizio. Un servizio può aggiornare queste politiche gestite con nuove autorizzazioni quando ne rilascia di nuove. APIs AWS X-Ray fornisce policy gestite per casi d'uso di sola lettura, sola scrittura e amministratore.

- `AWSXrayReadOnlyAccess`— Autorizzazioni di lettura per l'utilizzo della console X-Ray AWS o dell'SDK per ottenere dati di traccia AWS CLI, mappe di traccia, approfondimenti e configurazione X-Ray dall'API X-Ray. Include Observability Access Manager (OAM) `oam:ListSinks` e `oam:ListAttachedSinks` autorizzazioni per consentire alla console di visualizzare le tracce condivise dagli account di origine come parte dell'osservabilità tra account. CloudWatch Le azioni `BatchGetTraceSummaryById` e `GetDistinctTraceGraphs` API non sono pensate per essere richiamate dal codice e non sono incluse nel comando and. AWS CLI AWS SDKs

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries",
                "xray:BatchGetTraces",
                "xray:BatchGetTraceSummaryById",
                "xray:GetDistinctTraceGraphs",
                "xray:GetServiceGraph",
                "xray:GetTraceGraph",
                "xray:GetTraceSummaries",
                "xray:GetGroups",
                "xray:GetGroup",
                "xray:ListTagsForResource",
                "xray:ListResourcePolicies",
                "xray:GetTimeSeriesServiceStatistics",
                "xray:GetInsightSummaries",
                "xray:GetInsight",
                "xray:GetInsightEvents",
                "xray:GetInsightImpactGraph",
                "oam:ListSinks"
            ],
            "Resource": [
```

```
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }

}
```

- `AWSXRayDaemonWriteAccess`— Autorizzazioni di scrittura per l'utilizzo del demone X-Ray AWS o SDK per caricare AWS CLI documenti di segmento e telemetria nell'API X-Ray. Include le autorizzazioni di lettura per accedere alle regole di campionamento e i report sui risultati del campionamento.

  JSON

  ```
  {
      "Version":"2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                  "xray:PutTraceSegments",
                  "xray:PutTelemetryRecords",
                  "xray:GetSamplingRules",
                  "xray:GetSamplingTargets",
                  "xray:GetSamplingStatisticSummaries"
              ],
              "Resource": [
                  "*"
              ]
          }
      ]
  }
  ```

- `AWSXrayCrossAccountSharingConfiguration`— Concede le autorizzazioni per creare, gestire e visualizzare i collegamenti di Observability Access Manager per la condivisione di risorse X-Ray tra account. Utilizzato per consentire l'osservabilità CloudWatch tra account di origine e account di monitoraggio.

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`— Autorizzazione all'uso di tutti gli X-Ray APIs, inclusi i permessi di lettura, i permessi di scrittura e l'autorizzazione a configurare le impostazioni delle chiavi di crittografia e le regole di campionamento. Include Observability Access Manager (OAM) `oam:ListSinks` e `oam:ListAttachedSinks` autorizzazioni per consentire alla console di visualizzare le tracce condivise dagli account di origine come parte dell'osservabilità tra account. CloudWatch

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:*",
                "oam:ListSinks"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

Per aggiungere una policy gestita a un utente, gruppo o ruolo IAM

1. Apri la console IAM.
2. Aprire il ruolo associato al profilo dell'istanza, a un utente IAM o a un gruppo IAM.
3. In Permissions (Autorizzazioni), collegare la policy gestita.

Aggiornamenti X-Ray alle AWS policy gestite

Visualizza i dettagli sugli aggiornamenti delle policy AWS gestite per X-Ray da quando questo servizio ha iniziato a tracciare queste modifiche. Per ricevere avvisi automatici sulle modifiche apportate a questa pagina, abbonatevi al feed RSS nella pagina della cronologia dei documenti X-Ray.

Modifica	Descrizione	Data
Policy gestite da IAM per X-Ray: sono state aggiunte policy nuove `AWSXrayCrossAccountSharingConfiguration` `AWSXrayReadOnlyAccess` e `AWSXrayFullAccess` aggiornate.	X-Ray ha aggiunto le autorizzazioni di Observability Access Manager (OAM) `oam:ListSinks` e `oam:ListAttachedSinks` a queste policy per consentire alla console di visualizzare le tracce condivise dagli account di origine come parte dell'osservabilità tra account. CloudWatch	27 novembre 2022
Policy gestite da IAM per X-Ray: aggiornamento della policy. `AWSXrayReadOnlyAccess`	X-Ray ha aggiunto un'azione API,. `ListResourcePolicies`	15 novembre 2022
Utilizzo della console X-Ray: aggiornamento della policy `AWSXrayReadOnlyAccess`	X-Ray ha aggiunto due nuove azioni API e. `BatchGetTraceSummaryById` `GetDistinctTraceGraphs` Queste azioni non sono destinate a essere richiamate dal codice. Pertanto, queste azioni API non sono incluse in AWS CLI and AWS SDKs.	11 novembre 2022

Specificare una risorsa all'interno di una policy IAM

È possibile controllare l'accesso alle risorse utilizzando una policy IAM. Per le operazioni che supportano le autorizzazioni a livello di risorsa, si utilizza un Amazon Resource Name (ARN) per identificare la risorsa a cui si applica la policy.

Tutte le azioni X-Ray possono essere utilizzate in una policy IAM per concedere o negare agli utenti il permesso di utilizzare quell'azione. Tuttavia, non tutte le azioni X-Ray supportano le autorizzazioni a livello di risorsa, che consentono di specificare le risorse su cui è possibile eseguire un'azione.

Per le operazioni che non supportano le autorizzazioni a livello di risorsa, è necessario utilizzare "*" come risorsa.

Le seguenti azioni X-Ray supportano le autorizzazioni a livello di risorsa:

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

Di seguito è riportato un esempio di policy di autorizzazione basata su identità per un'operazione `CreateGroup`. L'esempio mostra l'uso di un ARN relativo al nome di gruppo `local-users` con l'ID univoco come carattere jolly. L'ID univoco viene generato quando il gruppo viene creato, quindi non può essere previsto nella policy in anticipo. Quando utilizzi `GetGroup`, `UpdateGroup` oppure `DeleteGroup`, è possibile definirlo come un carattere jolly o l'ARN esatto, incluso l'ID.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateGroup"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
            ]
        }
```

```
    ]
}
```

Di seguito è riportato un esempio di policy di autorizzazione basata su identità per un'operazione `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**
>
> L'ARN di una regola di campionamento è definito in base al nome. A differenza del gruppo ARNs, le regole di campionamento non hanno un ID generato in modo univoco.

Risoluzione dei problemi di AWS X-Ray identità e accesso

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con X-Ray e IAM.

Argomenti

- Non sono autorizzato a eseguire un'azione in X-Ray
- Non sono autorizzato a eseguire iam: PassRole

- Sono un amministratore e voglio consentire ad altri di accedere a X-Ray
- Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse X-Ray

Non sono autorizzato a eseguire un'azione in X-Ray

Se ti Console di gestione AWS dice che non sei autorizzato a eseguire un'azione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è colui che ti ha fornito le credenziali di accesso.

L'errore di esempio seguente si verifica quando l'`mateojackson`utente tenta di utilizzare la console per visualizzare i dettagli su una regola di campionamento ma non dispone delle `xray:GetSamplingRules` autorizzazioni.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: xray:GetSamplingRules on resource: arn:${Partition}:xray:${Region}:
${Account}:sampling-rule/${SamplingRuleName}
```

In questo caso, Mateo chiede al suo amministratore di aggiornare le sue policy per poter accedere alla risorsa di esempio relativa alla regola mediante l'operazione `xray:GetSamplingRules`.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue policy devono essere aggiornate per consentirti di trasferire un ruolo a X-Ray.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un utente IAM denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in X-Ray. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Sono un amministratore e voglio consentire ad altri di accedere a X-Ray

Per consentire ad altri di accedere a X-Ray, è necessario concedere l'autorizzazione alle persone o alle applicazioni che devono accedervi. Se si utilizza Centro identità AWS IAM per gestire persone e applicazioni, si assegnano set di autorizzazioni a utenti o gruppi per definirne il livello di accesso. I set di autorizzazioni creano e assegnano automaticamente le policy IAM ai ruoli IAM associati alla persona o all'applicazione. Per ulteriori informazioni, consulta Set di autorizzazioni nella Guida per l'Centro identità AWS IAM utente.

Se non utilizzi IAM Identity Center, devi creare entità IAM (utenti o ruoli) per le persone o le applicazioni che necessitano di accesso. È quindi necessario allegare una policy all'entità che concede loro le autorizzazioni corrette in X-Ray. Dopo aver concesso le autorizzazioni, fornite le credenziali all'utente o allo sviluppatore dell'applicazione. Utilizzeranno tali credenziali per accedere. AWSPer ulteriori informazioni sulla creazione di utenti, gruppi, policy e autorizzazioni IAM, consulta IAM Identities and Policies and permissions in IAM nella IAM User Guide.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse X-Ray

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se X-Ray supporta queste funzionalità, vedere. Come AWS X-Ray funziona con IAM
- Per scoprire come fornire l'accesso alle tue risorse su tutto Account AWS ciò che possiedi, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.

- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Registrazione e monitoraggio AWS X-Ray

Il monitoraggio è importante per garantire l'affidabilità, la disponibilità e le prestazioni delle soluzioni AWS . È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. AWS fornisce diversi strumenti per monitorare le risorse X-Ray e rispondere a potenziali incidenti:

AWS CloudTrail registri

AWS X-Ray si integra con AWS CloudTrail per registrare le azioni API eseguite da un utente, un ruolo o un AWS servizio in X-Ray. Puoi utilizzarlo CloudTrail per monitorare le richieste API X-Ray in tempo reale e archiviare i log in Amazon S3, Amazon Logs e Amazon Events. CloudWatch CloudWatch Per ulteriori informazioni, consulta Registrazione delle chiamate all'API X-Ray con AWS CloudTrail.

AWS Config Monitoraggio

AWS X-Ray si integra con AWS Config per registrare le modifiche alla configurazione apportate alle risorse di crittografia X-Ray. È possibile utilizzarlo AWS Config per inventariare le risorse di crittografia a raggi X, controllare la cronologia delle configurazioni a raggi X e inviare notifiche in base alle modifiche delle risorse. Per ulteriori informazioni, consulta Monitoraggio delle modifiche alla configurazione della X-Ray crittografia con AWS Config.

CloudWatch Monitoraggio Amazon

Puoi utilizzare X-Ray SDK for Java per pubblicare metriche CloudWatch Amazon non campionate dai segmenti X-Ray raccolti. Questi parametri sono derivati dall'ora di inizio e di fine del segmento e dai flag di errore, di malfunzionamento e di stato trottled. Utilizzare questi parametri di tracciamento per esporre tentativi e problemi di dipendenza all'interno di sottosegmenti. Per ulteriori informazioni, consulta AWS X-Ray metriche per X-Ray SDK for Java.

Convalida della conformità per AWS X-Ray

La nostra nuova esperienza AWS di iscrizione non è progettata per carichi di lavoro regolamentati. Se stai utilizzando la nostra nuova esperienza di AWS registrazione, ma desideri utilizzarla AWS per carichi di lavoro regolamentati, puoi iscriverti a AWS (avanzato) o attivare funzionalità avanzate per il tuo ambiente. AWS

Per sapere se un Servizio AWS rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program e scegli il programma di conformità che ti interessa. Per informazioni generali, vedere Programmi di AWS conformità Programmi di

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, vedere Scaricamento dei report in AWS Artifact .

La responsabilità dell'utente in materia di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità dell'utente in materia di conformità durante l'utilizzo Servizi AWS, consulta AWS la Documentazione sulla sicurezza.

Resilienza in AWS X-Ray

L'infrastruttura AWS globale è costruita attorno a Regioni AWS zone di disponibilità. Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate, collegate con reti a bassa latenza, ad alto throughput e altamente ridondanti. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni sulle zone di disponibilità, vedere Global Regioni AWS Infrastructure.AWS

Sicurezza dell'infrastruttura in AWS X-Ray

In quanto servizio gestito, AWS X-Ray è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Si utilizzano chiamate API AWS pubblicate per accedere a X-Ray attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Utilizzo AWS X-Ray con endpoint VPC

Se utilizzi Amazon Virtual Private Cloud (Amazon VPC) per ospitare AWS le tue risorse, puoi stabilire una connessione privata tra il tuo VPC e. X-Ray Ciò consente alle risorse del tuo Amazon VPC di comunicare con il X-Ray servizio senza passare attraverso la rete Internet pubblica.

Amazon VPC è un programma Servizio AWS che puoi utilizzare per avviare AWS risorse in una rete virtuale da te definita. Con un VPC, detieni il controllo delle impostazioni della rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. Per connettere il tuo VPC a X-Ray, definisci un endpoint VPC di interfaccia. L'endpoint fornisce una connettività affidabile e scalabile X-Ray senza richiedere un gateway Internet, un'istanza NAT (Network Address Translation) o una connessione VPN. Per ulteriori informazioni, consulta Che cos'è Amazon VPC? nella Guida per l'utente Amazon VPC.

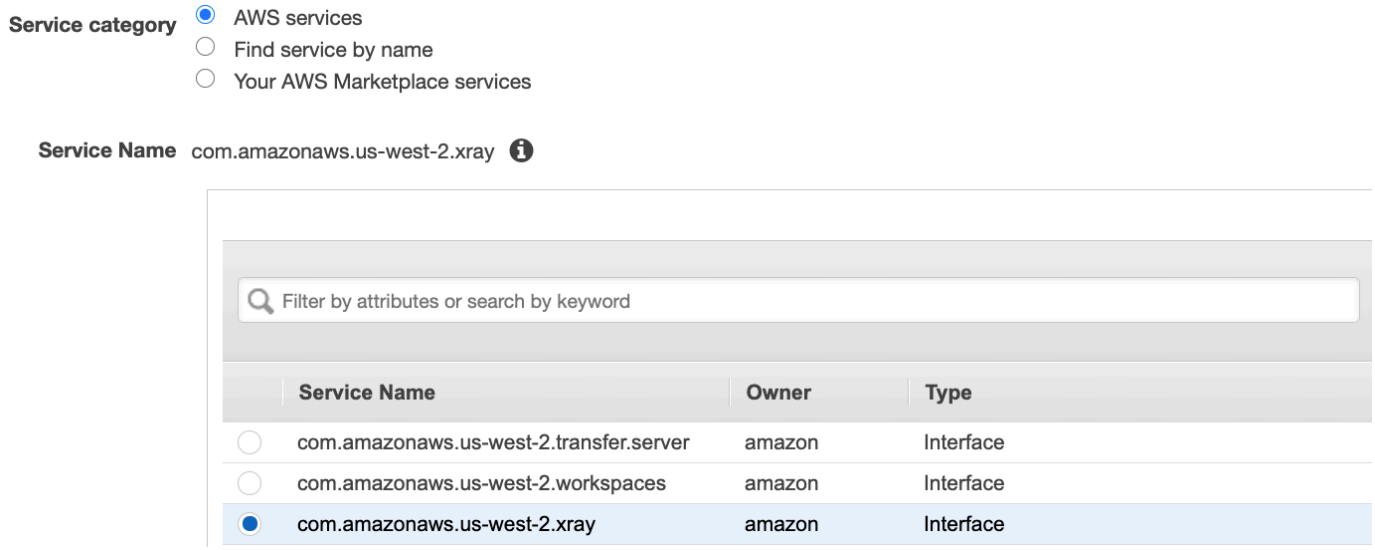
Gli endpoint Interface VPC sono alimentati da AWS PrivateLink, una AWS tecnologia che consente la comunicazione privata tra loro Servizi AWS utilizzando un'interfaccia di rete elastica con indirizzi IP privati. Per ulteriori informazioni, consulta il post del Servizi AWS blog New — AWS PrivateLink for e Getting Started in the Amazon VPC User Guide.

Per assicurarti di poter creare un endpoint VPC per X-Ray quello che hai scelto Regione AWS, consulta. Regioni supportate

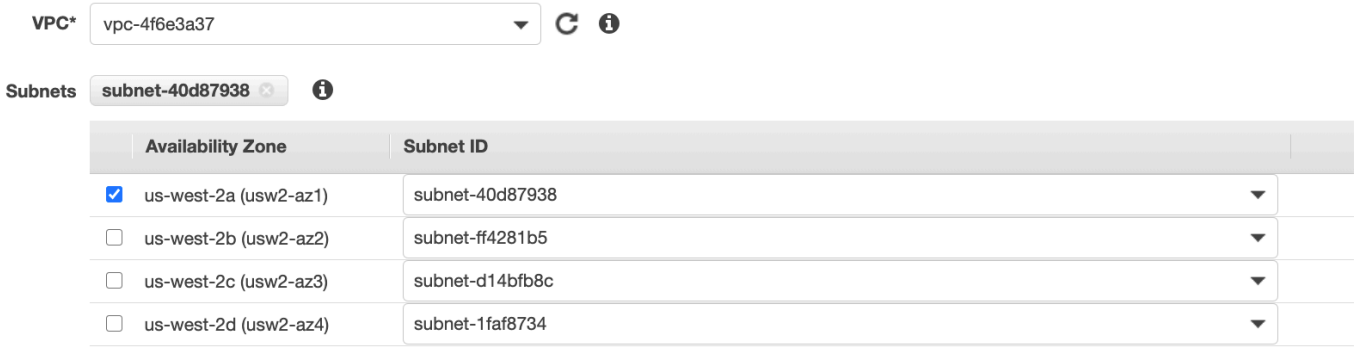
Creazione di un endpoint VPC per X-Ray

Per iniziare a utilizzarlo X-Ray con il tuo VPC, crea un endpoint VPC di interfaccia per. X-Ray

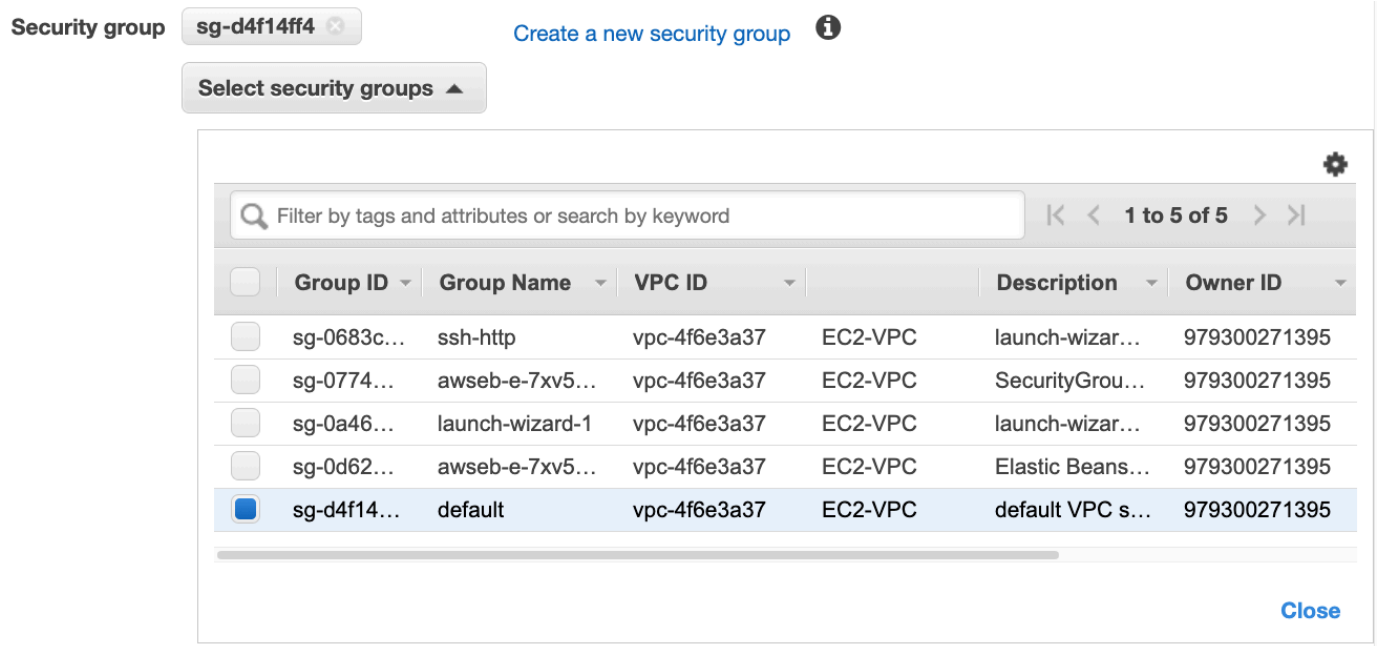
1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Accedi a Endpoint nel pannello di navigazione e scegli Create Endpoint.
3. Cerca e seleziona il nome del AWS X-Ray servizio:. `com.amazonaws.region.xray`



4. Seleziona il VPC che desideri, quindi seleziona una sottorete nel tuo VPC per utilizzare l'endpoint dell'interfaccia. Un'interfaccia di rete endpoint viene creata nella sottorete selezionata. Puoi specificare più sottoreti in zone di disponibilità differenti (come supportato dal servizio) per garantire che l'endpoint di interfaccia sia resiliente a errori della zona di disponibilità. In tal caso, viene creata un'interfaccia di rete in ogni sottorete specificata.



5. (Facoltativo) Il DNS privato è abilitato per impostazione predefinita per l'endpoint, in modo da poter effettuare richieste X-Ray utilizzando il nome host DNS predefinito. Puoi scegliere di disabilitarlo.
6. Specifica i gruppi di sicurezza da associare all'interfaccia di rete dell'endpoint.



7. (Facoltativo) Specifica una politica personalizzata per controllare le autorizzazioni di accesso al X-Ray servizio. Per impostazione predefinita, è consentito l'accesso completo.

Controllo dell'accesso al tuo X-Ray endpoint VPC

Una policy endpoint VPC è una policy della risorsa IAM che viene collegata a un endpoint durante la creazione o la modifica dell'endpoint. Se non colleghi una policy durante la creazione di un endpoint, Amazon VPC collega una policy predefinita che consente l'accesso completo al servizio. Una policy endpoint non esclude né sostituisce policy dell'utente IAM o policy specifiche del servizio. Si tratta di una policy separata per controllare l'accesso dall'endpoint al servizio specificato. Le policy endpoint devono essere scritte in formato JSON. Per ulteriori informazioni, consultare Controllo degli accessi ai servizi con endpoint VPC nella Guida per l'utente di Amazon VPC.

La policy sugli endpoint VPC consente di controllare le autorizzazioni per varie azioni. X-Ray Ad esempio, puoi creare una policy per consentire solo PutTraceSegment e negare tutte le altre azioni. Ciò limita i carichi di lavoro e i servizi nel VPC a inviare solo dati di traccia X-Ray e a negare qualsiasi altra azione come il recupero dei dati, la modifica della configurazione di crittografia o i gruppi. create/update

Di seguito è riportato un esempio di policy sugli endpoint per. X-Ray Questa policy consente agli utenti che si connettono X-Ray tramite il VPC di inviare dati di segmento a X-Ray e impedisce loro di eseguire altre X-Ray azioni.

```
{"Statement": [
    {"Sid": "Allow PutTraceSegments",
      "Principal": "*",
      "Action": [
        "xray:PutTraceSegments"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Per modificare la politica degli endpoint VPC per X-Ray

1. Apri la console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel pannello di navigazione, seleziona Endpoint.
3. Se non hai già creato l'endpoint per X-Ray, segui i passaggi indicati. Creazione di un endpoint VPC per X-Ray
4. Seleziona com.amazonaws. ***region***.xray endpoint, quindi scegli la scheda Policy.
5. Scegli Edit Policy (Modifica policy), quindi apporta le modifiche.

Regioni supportate

X-Ray attualmente supporta i seguenti endpoint VPC: Regioni AWS

- Stati Uniti orientali (Ohio)
- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacifico (Osaka)
- Asia Pacifico (Seoul)
- Asia Pacifico (Singapore)

- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Europa (Francoforte)
- Europa (Irlanda)
- Europa (Londra)
- Europa (Milano)
- Europa (Parigi)
- Europa (Stoccolma)
- Medio Oriente (Bahrein)
- Sud America (San Paolo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Prevenzione del confused deputy tra servizi

Il problema confused deputy è un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire un'azione può costringere un'entità maggiormente privilegiata a eseguire l'azione. Nel AWS, l'impersonificazione tra servizi può portare al confuso problema del vice. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse dell'account.

Si consiglia di utilizzare le chiavi di contesto `aws:SourceArn`aws:SourceAccount, `aws:SourceOrgID`, e `aws:SourceOrgPaths`global condition nelle politiche delle risorse per limitare i permessi che xraylong concede a un altro servizio alla risorsa. Utilizza `aws:SourceArn` per associare una sola risorsa all'accesso tra servizi. Utilizza `aws:SourceAccount` se desideri consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi. Utilizza `aws:SourceOrgID` se desideri consentire l'associazione di qualsiasi risorsa di qualsiasi account interno a un'organizzazione all'uso tra servizi. Utilizza `aws:SourceOrgPaths` per associare qualsiasi risorsa dagli account in un percorso AWS Organizations all'uso tra servizi. Per ulteriori

informazioni sull'uso e la comprensione dei percorsi, consulta Comprendere il percorso dell' AWS Organizations entità.

Il modo più efficace per proteggersi dal problema “confused deputy” è quello di utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con l’ARN completo della risorsa. Se non si conosce l’ARN completo della risorsa o si scelgono più risorse, utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell’ARN. Ad esempio, `arn:aws:servicename:*:123456789012:*`.

Se il valore `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare sia `aws:SourceAccount` che `aws:SourceArn` per limitare le autorizzazioni.

Per proteggersi dal problema "confused deputy" su larga scala, nelle policy basate sulle risorse utilizza la chiave di contesto della condizione globale `aws:SourceOrgID` o `aws:SourceOrgPaths` con l'ID dell'organizzazione o il percorso dell'organizzazione della risorsa. Quando aggiungi, rimuovi o sposti degli account all'interno dell'organizzazione, le policy che includono la chiave `aws:SourceOrgID` o `aws:SourceOrgPaths` includono automaticamente anche gli account corretti e non necessitano dell'aggiornamento manuale.

L'esempio seguente mostra come utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` global condition in xray per prevenire il confuso problema del vice.

```
{
      "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:*"
```

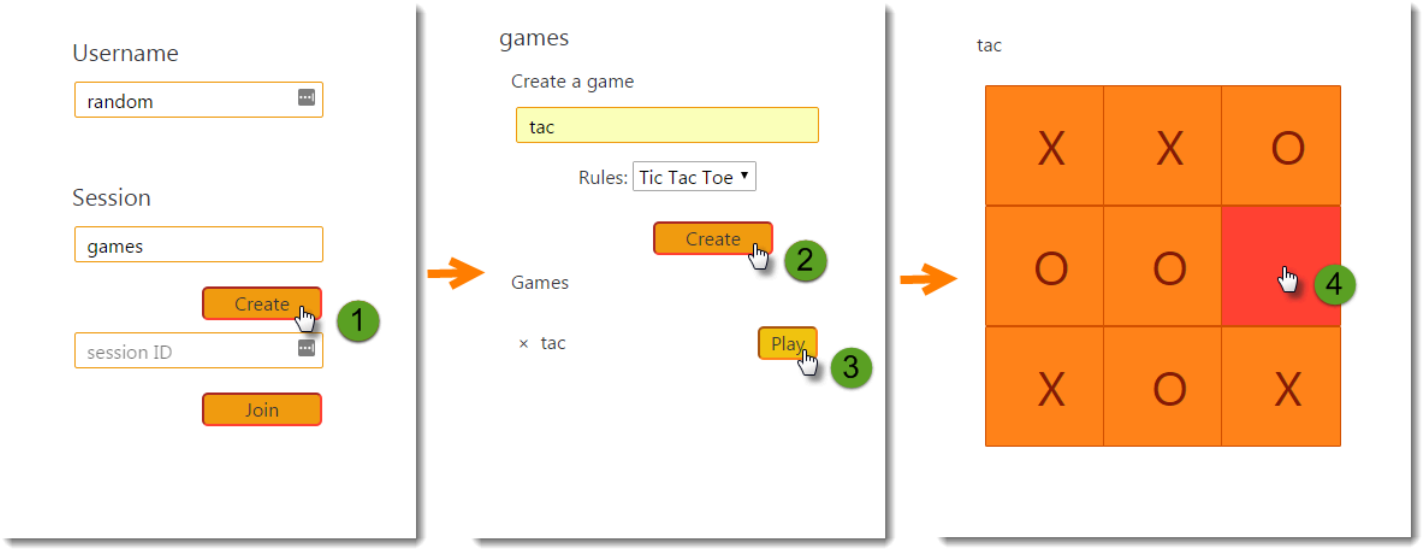
```
        }
      }
    }
```

AWS X-Ray applicazione di esempio

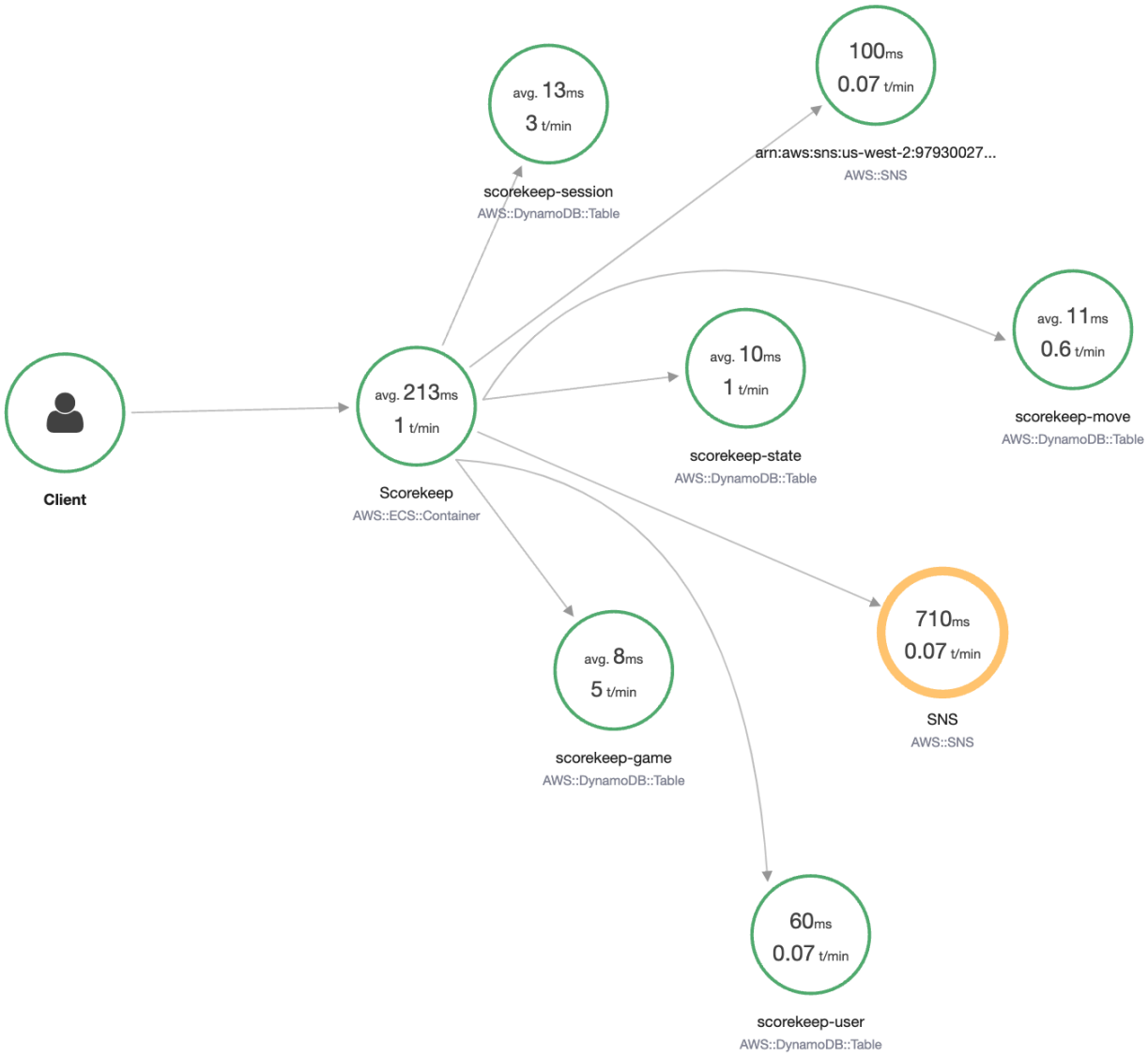
> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione — Il 25 febbraio 2026 AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, che AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

L'app di esempio AWS X-Ray eb-java-scorekeep, disponibile su, mostra l'uso dell'SDK per monitorare le chiamate HTTP in entrata GitHub, i client DynamoDB AWS X-Ray SDK e i client HTTP. L'app di esempio utilizza CloudFormation per creare tabelle DynamoDB, compilare codice Java su istanza ed eseguire il demone senza alcuna configurazione aggiuntiva. X-Ray

Consulta il tutorial di Scorekeep per iniziare a installare e utilizzare un'applicazione di esempio strumentata, utilizzando o. Console di gestione AWS AWS CLI



L'esempio include un'app Web front-end, l'API che chiama e le tabelle DynamoDB utilizzate per archiviare i dati. La strumentazione di base con filtri, plugin e client AWS SDK strumentati è mostrata nel ramo del progetto. `xray-gettingstarted` Questo è il ramo che viene distribuito nel tutorial sulle nozioni di base. Poiché questo ramo include solo le nozioni di base, puoi eseguire una diff rispetto al ramo `master` per individuare rapidamente le nozioni di base.



L'applicazione di esempio illustra l'analisi di base in questi file:

- Filtro di richiesta HTTP — `WebConfig.java`
- AWS Strumentazione client SDK — `build.gradle`

Il `xray` ramo dell'applicazione include l'uso di HttpClient, Annotations, query SQL, sottosegmenti personalizzati , una funzione strumentata e codice e script di inizializzazione strumentati AWS Lambda. ???

Per supportare l'accesso e l' AWS SDK per JavaScript utilizzo degli utenti nel browser, la `xray-cognito` filiale aggiunge Amazon Cognito per supportare l'autenticazione e l'autorizzazione

degli utenti. Grazie alle credenziali recuperate da Amazon Cognito, l'app Web invia anche dati di tracciamento per X-Ray registrare le informazioni sulla richiesta dal punto di vista del cliente. Il client del browser appare come nodo a sé stante sulla mappa di tracciamento e registra informazioni aggiuntive, tra cui l'URL della pagina che l'utente sta visualizzando e l'ID dell'utente.

Infine, la `xray-worker` filiale aggiunge una funzione Python Lambda strumentata che viene eseguita in modo indipendente, elaborando gli elementi da una coda Amazon SQS. Scorekeep aggiunge un elemento alla coda ogni volta che termina una partita. Il worker Lambda, attivato da CloudWatch Events, estrae gli elementi dalla coda ogni pochi minuti e li elabora per archiviare i record di gioco in Amazon S3 per l'analisi.

Argomenti

- Guida introduttiva all'applicazione di esempio Scorekeep
- Strumentazione manuale dei client SDK AWS
- Creazione di sottosegmenti aggiuntivi
- Registrazione di annotazioni, metadati e ID utente
- Analisi delle chiamate HTTP in uscita
- Analizzare le chiamate a un database PostgresSQL
- • Strumentazione AWS Lambda funzioni
- Analisi del codice di avvio
- Analisi degli script
- Analisi di un client di app Web
- Utilizzo dei client analizzati nei thread worker

Guida introduttiva all'applicazione di esempio Scorekeep

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, che AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

Questo tutorial utilizza il `xray-gettingstarted` ramo dell'applicazione di esempio Scorekeep, utilizzato CloudFormation per creare e configurare le risorse che eseguono l'applicazione e il demone di esempio su Amazon ECS. X-Ray L'applicazione utilizza il framework Spring per implementare un'API web JSON e per rendere persistenti i dati in Amazon AWS SDK per Java DynamoDB. Un filtro servlet nell'applicazione gestisce tutte le richieste in entrata servite dall'applicazione e un gestore di richieste sul client AWS SDK gestisce le chiamate a valle verso DynamoDB.

Puoi seguire questo tutorial usando il o il. Console di gestione AWS AWS CLI

Sections

- Prerequisiti
- Installa l'applicazione Scorekeep utilizzando CloudFormation
- Generare i dati di tracciamento
- Visualizza la mappa delle tracce nel Console di gestione AWS
- Configurazione delle notifiche Amazon SNS
- Esplorare l'applicazione di esempio
- Opzionale: policy con privilegio minimo
- Eliminazione
- Fasi successive

Prerequisiti

Questo tutorial serve CloudFormation a creare e configurare le risorse che eseguono l'applicazione e il X-Ray demone di esempio. I seguenti prerequisiti sono necessari per installare ed eseguire il tutorial:

1. Se utilizzi un utente IAM con autorizzazioni limitate, aggiungi le seguenti politiche utente nella console IAM:

   - `AWSCloudFormationFullAccess`— per accedere e utilizzare CloudFormation
   - `AmazonS3FullAccess`— per caricare un file modello da CloudFormation utilizzare Console di gestione AWS
   - `IAMFullAccess`— per creare i ruoli delle istanze Amazon ECS e Amazon EC2
   - `AmazonEC2FullAccess`— per creare le risorse Amazon EC2
   - `AmazonDynamoDBFullAccess`— per creare le tabelle DynamoDB

- `AmazonECS_FullAccess`— per creare risorse Amazon ECS
- `AmazonSNSFullAccess`— per creare l'argomento Amazon SNS
- `AWSXrayReadOnlyAccess`— per ottenere il permesso di visualizzare la mappa delle tracce e le tracce nella console X-Ray

2. Per eseguire il tutorial utilizzando il AWS CLI, installa la CLI versione 2.7.9 o successiva e configura la CLI con l'utente del passaggio precedente. Assicurati che la regione sia configurata durante la configurazione con l'utente. AWS CLI Se una regione non è configurata, dovrai aggiungerla `--region` *`AWS-REGION`* a ogni comando CLI.
3. Assicurati che Git sia installato, per clonare il repository dell'applicazione di esempio.
4. Usa il seguente esempio di codice per clonare il `xray-gettingstarted` ramo del repository Scorekeep:

```
git clone https://github.com/aws-samples/eb-java-scorekeep.git xray-scorekeep -b
 xray-gettingstarted
```

Installa l'applicazione Scorekeep utilizzando CloudFormation

Console di gestione AWS

Installa l'applicazione di esempio utilizzando Console di gestione AWS

1. Apri la console CloudFormation
2. Scegli Crea stack, quindi scegli Con nuove risorse dal menu a discesa.
3. Nella sezione Specify template(Specifica il modello) scegliere Upload a template file (Carica un file modello).
4. Seleziona Scegli file, vai alla `xray-scorekeep/cloudformation` cartella creata quando hai clonato il repository git e scegli il file. `cf-resources.yaml`
5. Seleziona Successivo per continuare.
6. Entra `scorekeep` nella casella di testo Nome dello stack, quindi scegli Avanti nella parte inferiore della pagina per continuare. Nota che il resto di questo tutorial presuppone che lo stack abbia un nome. `scorekeep`
7. Scorri fino alla fine della pagina delle opzioni di configurazione dello stack e scegli Avanti per continuare.

8. Scorri fino alla fine della pagina di revisione, seleziona la casella di controllo che riconosce che CloudFormation può creare risorse IAM con nomi personalizzati e scegli Crea stack.
9. Lo CloudFormation stack è ora in fase di creazione. Lo stato dello stack durerà `CREATE_IN_PROGRESS` circa cinque minuti prima di passare a`CREATE_COMPLETE`. Lo stato verrà aggiornato periodicamente oppure puoi aggiornare la pagina.

AWS CLI

Installa l'applicazione di esempio utilizzando AWS CLI

1. Vai alla `cloudformation` cartella del `xray-scorekeep` repository che hai clonato in precedenza nel tutorial:

```
cd xray-scorekeep/cloudformation/
```

2. Inserisci il seguente AWS CLI comando per creare lo stack: CloudFormation

```
aws cloudformation create-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

3. Attendi lo stato dello CloudFormation stack`CREATE_COMPLETE`, operazione che richiederà circa cinque minuti. Usa il seguente AWS CLI comando per controllare lo stato:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Generare i dati di tracciamento

L'applicazione di esempio include un app web di front-end. Utilizza l'app web per generare traffico verso l'API e inviare dati di tracciamento verso X-Ray. Innanzitutto, recupera l'URL dell'app Web utilizzando Console di gestione AWS o: AWS CLI

Console di gestione AWS

Trova l'URL dell'applicazione utilizzando Console di gestione AWS

1. Apri la console CloudFormation
2. Scegli lo `scorekeep` stack dall'elenco.

3. Scegli la scheda Output nella pagina dello `scorekeep` stack e scegli il link `LoadBalancerUrl` URL per aprire l'applicazione web.

AWS CLI

Trova l'URL dell'applicazione utilizzando il AWS CLI

1. Utilizzate il seguente comando per visualizzare l'URL dell'applicazione web:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].Outputs[0].OutputValue"
```

2. Copia questo URL e aprilo in un browser per visualizzare l'applicazione web Scorekeep.

Usa l'applicazione web per generare dati di tracciamento

1. Scegliere Create (Crea) per creare un utente e una sessione.
2. Digitare un game name (nome del gioco), impostare Rules (Regole) a Tic Tac Toe (Tris) e quindi scegliere Create (Crea) per creare un gioco.
3. Scegliere Play (Gioca) per iniziare il gioco.
4. Scegliere un riquadro per fare una mossa e modificare lo stato del gioco.

Ognuno di questi passaggi genera richieste HTTP all'API e chiamate downstream a DynamoDB per leggere e scrivere dati relativi a utenti, sessioni, giochi, spostamenti e stati.

Visualizza la mappa delle tracce nel Console di gestione AWS

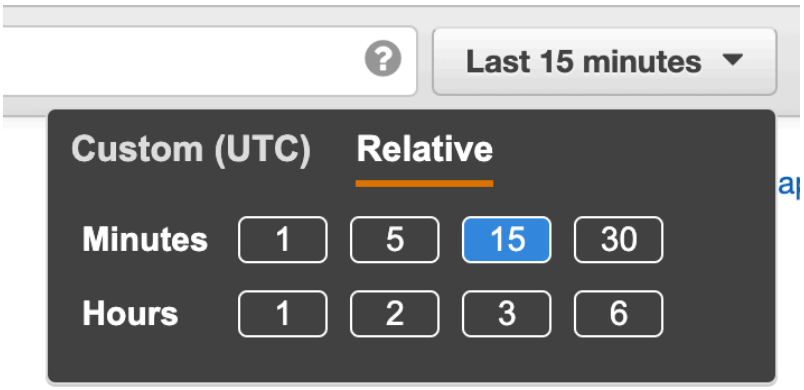
È possibile visualizzare la mappa delle tracce e le tracce generate dall'applicazione di esempio nelle CloudWatch console X-Ray e.

X-Ray console

Usa la console X-Ray

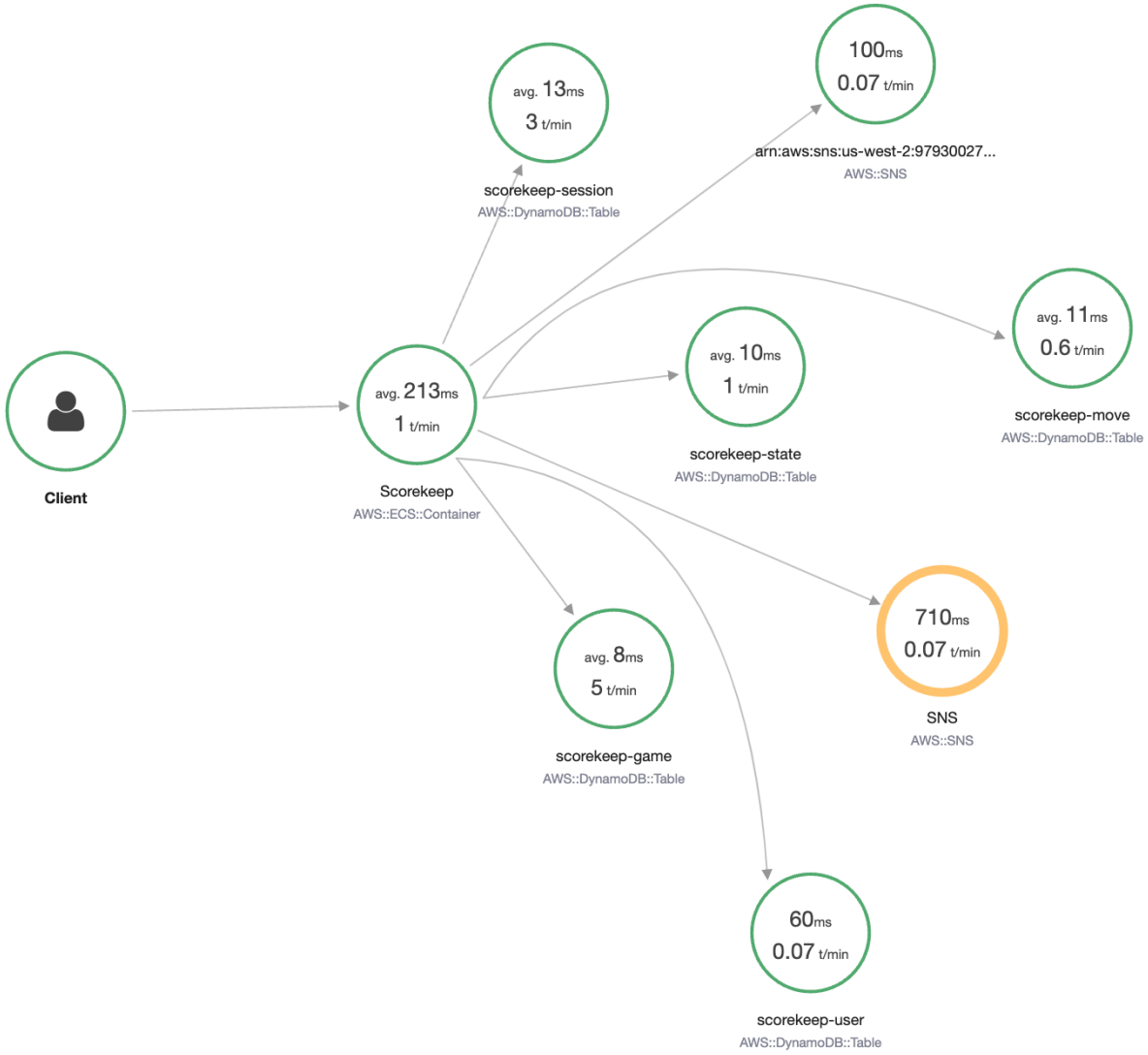
1. Aprire la pagina della mappa di tracciamento della X-Ray console.
2. La console mostra una rappresentazione del grafico del servizio X-Ray generato dai dati di traccia inviati dall'applicazione. Assicurati di modificare il periodo di tempo della mappa

di tracciamento, se necessario, per assicurarti che visualizzi tutte le tracce dal primo avvio dell'applicazione web.



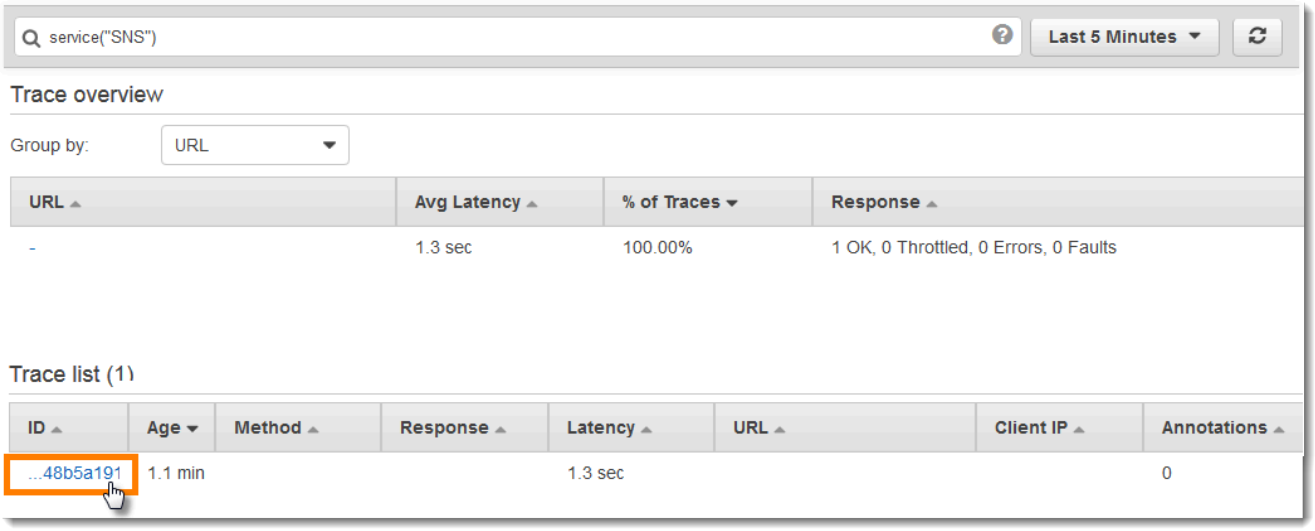
La mappa di tracciamento mostra il client dell'app Web, l'API in esecuzione in Amazon ECS e ogni tabella DynamoDB utilizzata dall'applicazione. Ogni richiesta all'applicazione, fino a un numero massimo configurabile di richieste al secondo, viene tracciata non appena viene ricevuta dall'API, genera le richieste verso i servizi a valle e si conclude.

Puoi scegliere qualsiasi nodo del grafo del servizio per visualizzare i dati di tracciamento per le richieste che hanno generato il traffico verso tale nodo. Attualmente, il nodo Amazon SNS è giallo. Approfondisci l'esplorazione per scoprire perché.

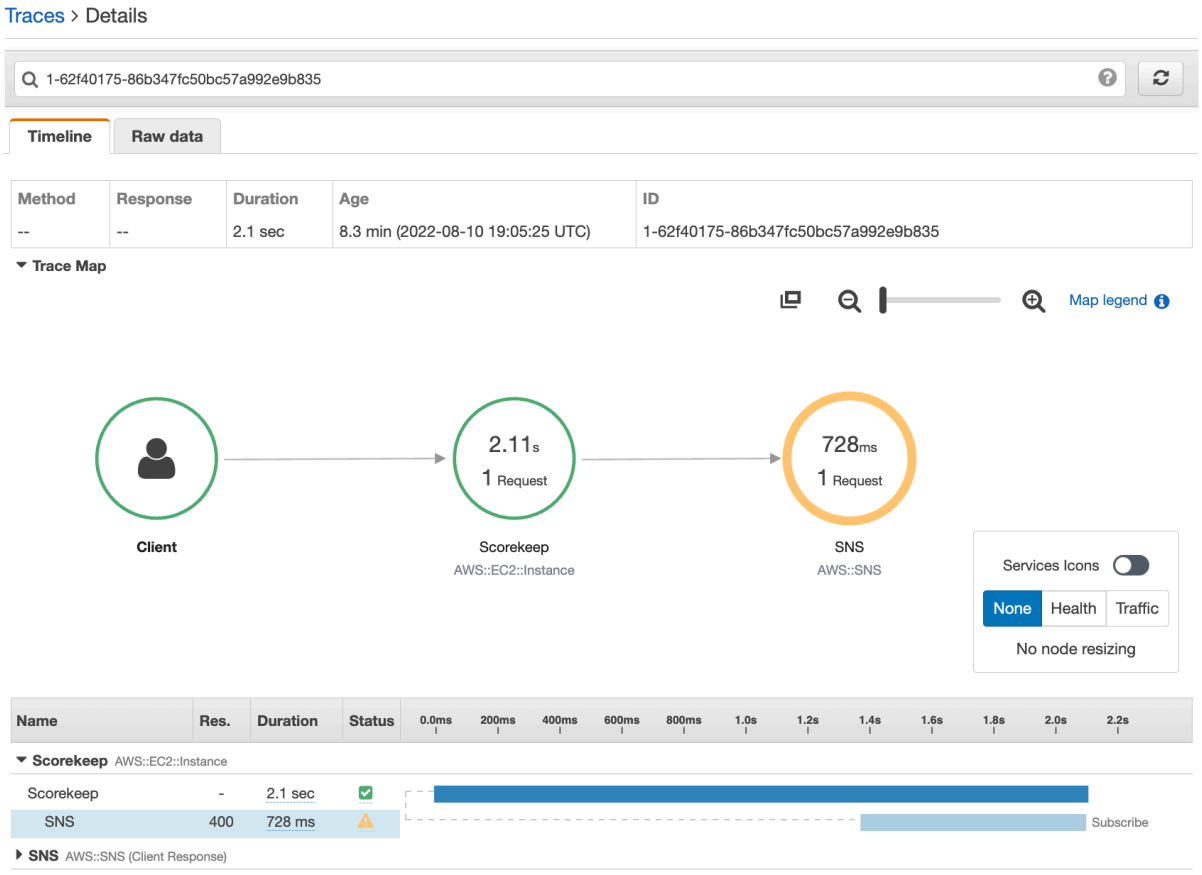


Per trovare la causa dell'errore

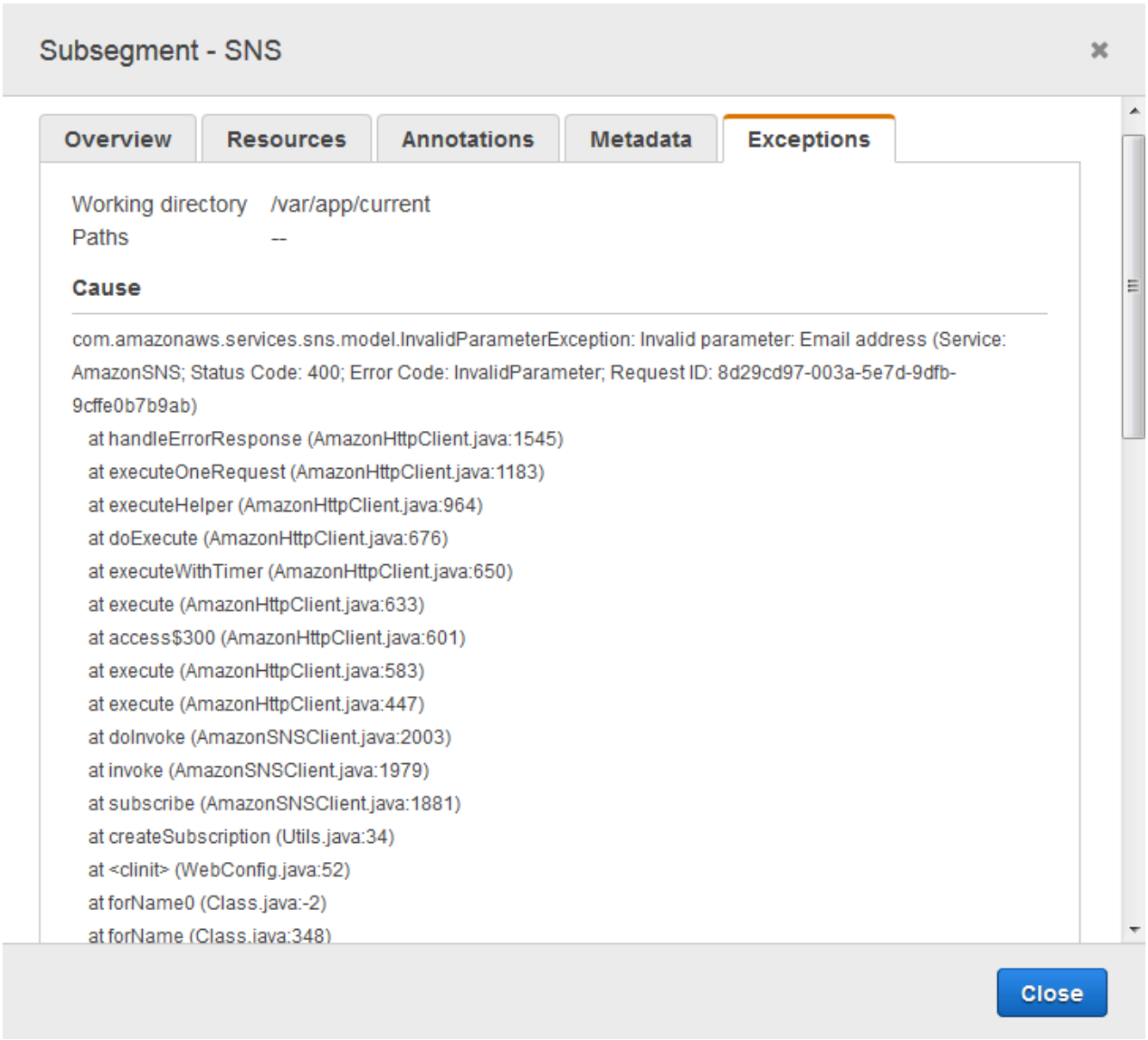
1. Scegliere il nodo denominato SNS. Viene visualizzato il pannello dei dettagli del nodo.
2. Scegliere View traces (Visualizza tracciamenti) per accedere alla schermata Trace overview (Panoramica tracciamenti).
3. Scegliere il tracciamento dalla Trace list (Elenco tracciamenti). A questo tracciamento non sono associati metodo o URL perché è stato registrato durante l'avvio anziché in risposta a una richiesta in entrata.



4. Scegli l'icona dello stato di errore all'interno del segmento Amazon SNS nella parte inferiore della pagina, per aprire la pagina delle eccezioni per il sottosegmento SNS.



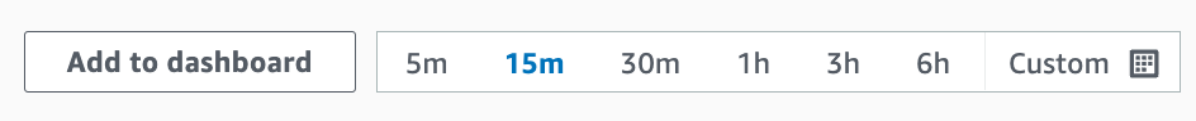
5. L' X-Ray SDK acquisisce automaticamente le eccezioni generate dai client SDK AWS strumentati e registra lo stack trace.



CloudWatch console

CloudWatch Usa la console

1. Aprire la pagina della mappa di X-Ray tracciamento della CloudWatch console.
2. La console mostra una rappresentazione del grafico del servizio X-Ray generato dai dati di traccia inviati dall'applicazione. Assicurati di modificare il periodo di tempo della mappa di tracciamento, se necessario, per assicurarti che visualizzi tutte le tracce dal primo avvio dell'applicazione web.



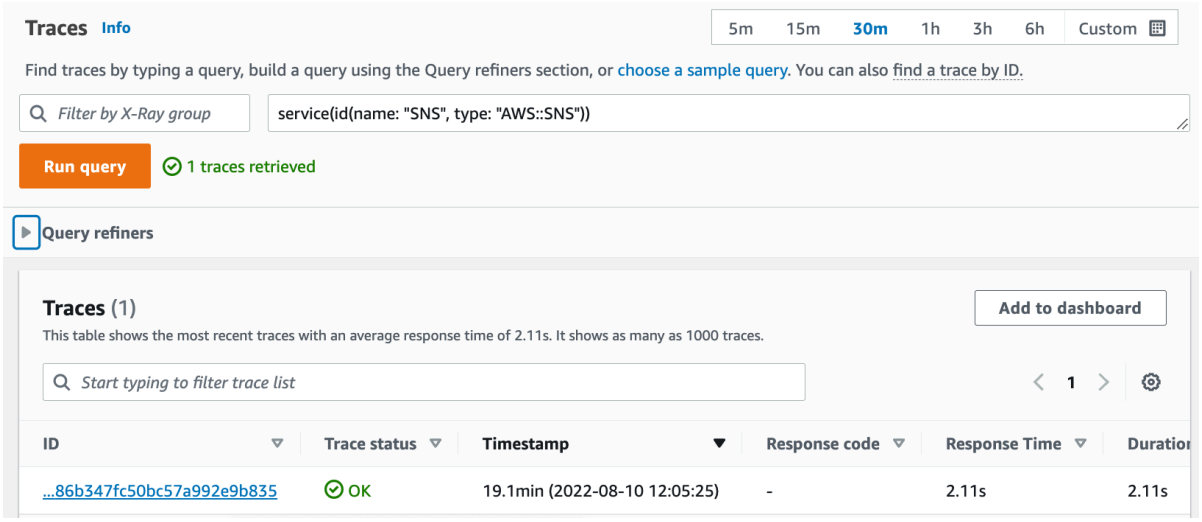
La mappa di tracciamento mostra il client dell'app Web, l'API in esecuzione in Amazon EC2 e ogni tabella DynamoDB utilizzata dall'applicazione. Ogni richiesta all'applicazione, fino a un numero massimo configurabile di richieste al secondo, viene tracciata non appena viene ricevuta dall'API, genera le richieste verso i servizi a valle e si conclude.

Puoi scegliere qualsiasi nodo del grafo del servizio per visualizzare i dati di tracciamento per le richieste che hanno generato il traffico verso tale nodo. Attualmente, il nodo Amazon SNS è arancione. Approfondisci l'esplorazione per scoprire perché.

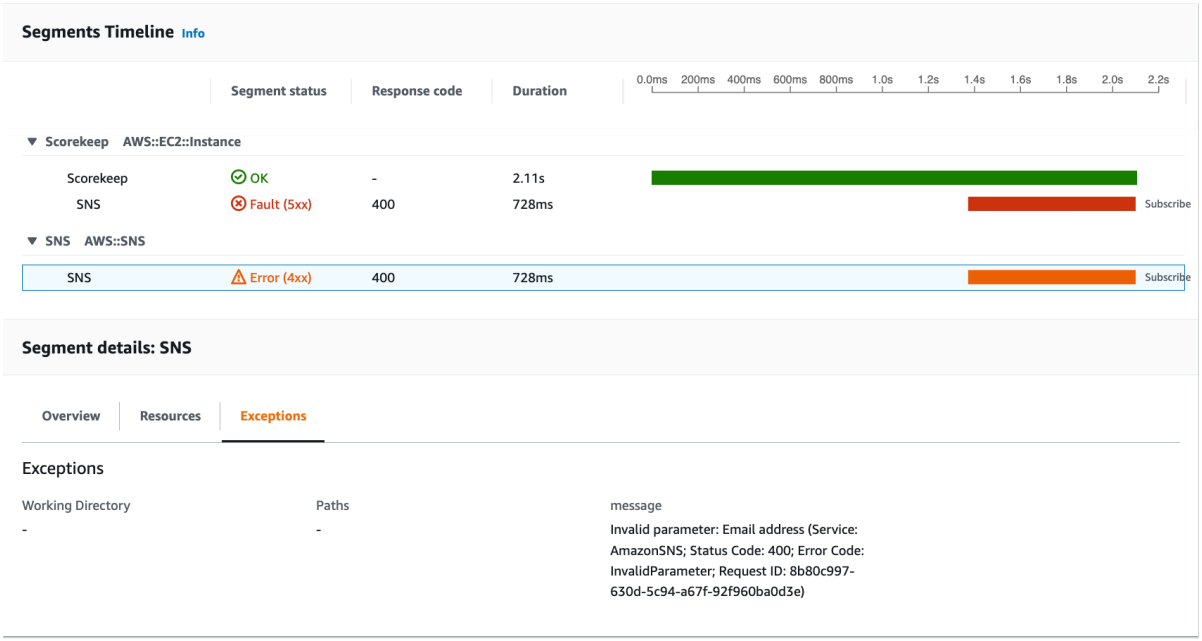

scorekeep-state
DynamoDB Table
scorekeep-game
DynamoDB Table
scorekeep-move
DynamoDB Table
Client
Scorekeep
ECS Container
SNS
SNS
scorekeep-session
DynamoDB Table

Per trovare la causa dell'errore

1. Scegliere il nodo denominato SNS. Il pannello dei dettagli del nodo SNS viene visualizzato sotto la mappa.
2. Scegli Visualizza tracce per accedere alla pagina Tracce.
3. Aggiungi la parte inferiore della pagina, scegli la traccia dall'elenco Tracce. A questo tracciamento non sono associati metodo o URL perché è stato registrato durante l'avvio anziché in risposta a una richiesta in entrata.



4. Scegli il sottosegmento Amazon SNS nella parte inferiore della timeline dei segmenti e scegli la scheda Eccezioni per il sottosegmento SNS per visualizzare i dettagli delle eccezioni.



La causa indica che l'indirizzo e-mail fornito in una chiamata a `createSubscription` invocata nella classe `WebConfig` non è valido. Nella prossima sezione, risolveremo il problema.

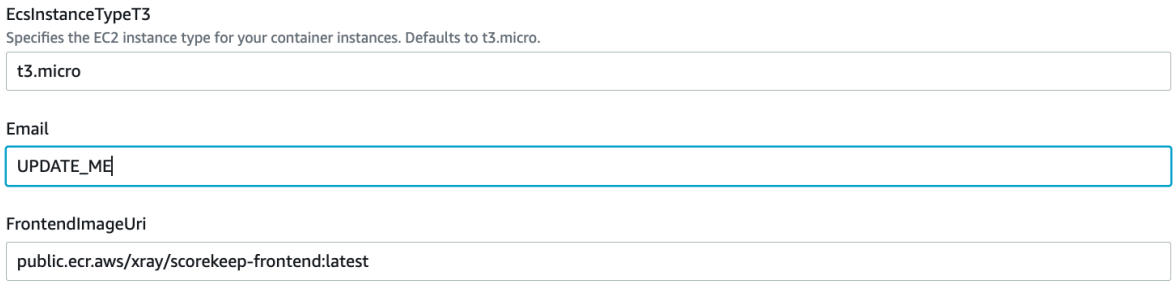
Configurazione delle notifiche Amazon SNS

Scorekeep utilizza Amazon SNS per inviare notifiche quando gli utenti completano una partita. Quando l'applicazione si avvia, tenta di creare un abbonamento per un indirizzo email definito in un CloudFormation parametro dello stack. Quella chiamata non riesce al momento. Configura un'email di notifica per abilitare le notifiche e risolvere gli errori evidenziati nella mappa di tracciamento.

Console di gestione AWS

Per configurare le notifiche Amazon SNS utilizzando Console di gestione AWS

1. Apri la console CloudFormation
2. Scegli il pulsante di opzione accanto al nome `scorekeep` dello stack nell'elenco, quindi scegli Aggiorna.
3. Assicurati che sia selezionato Usa il modello corrente, quindi fai clic su Avanti nella pagina Aggiorna stack.
4. Trova il parametro Email nell'elenco e sostituisci il valore predefinito con un indirizzo email valido.



5. Scorri fino alla parte inferiore della pagina e scegli Next (Avanti).
6. Scorri fino alla fine della pagina di revisione, seleziona la casella di controllo che riconosce che CloudFormation può creare risorse IAM con nomi personalizzati e scegli Aggiorna stack.
7. Lo CloudFormation stack è ora in fase di aggiornamento. Lo stato dello stack durerà `UPDATE_IN_PROGRESS` circa cinque minuti prima di passare a`UPDATE_COMPLETE`. Lo stato verrà aggiornato periodicamente oppure puoi aggiornare la pagina.

AWS CLI

Per configurare le notifiche di Amazon SNS utilizzando AWS CLI

1. Accedi alla `xray-scorekeep/cloudformation/` cartella che hai creato in precedenza e apri il `cf-resources.yaml` file in un editor di testo.
2. Trova il `Default` valore all'interno del parametro Email e modificalo *`UPDATE_ME`* in un indirizzo email valido.

```
Parameters:
  Email:
    Type: String
    Default: UPDATE_ME # <- change to a valid abc@def.xyz email address
```

3. Dalla `cloudformation` cartella, aggiorna lo CloudFormation stack con il seguente AWS CLI comando:

```
aws cloudformation update-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

4. Attendi lo stato dello CloudFormation stack`UPDATE_COMPLETE`, operazione che richiederà alcuni minuti. Usa il seguente AWS CLI comando per controllare lo stato:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Quando l'aggiornamento viene completato, Scorekeep si riavvia e crea una sottoscrizione per l'argomento SNS. Controlla la tua e-mail e conferma che la sottoscrizione per visualizzare gli aggiornamenti al completamento di un gioco. Apri la mappa di tracciamento per verificare che le chiamate a SNS non abbiano più esito negativo.

Esplorare l'applicazione di esempio

L'applicazione di esempio è un'API Web HTTP in Java configurata per utilizzare l' X-Ray SDK per Java. Quando si distribuisce l'applicazione con il CloudFormation modello, vengono create le tabelle DynamoDB, il cluster Amazon ECS e altri servizi necessari per eseguire Scorekeep su ECS. Un file di definizione delle attività per ECS viene creato tramite. CloudFormation Questo file definisce le immagini dei contenitori utilizzate per attività in un cluster ECS. Queste immagini sono ottenute dall'ECR X-Ray pubblico ufficiale. L'immagine del contenitore dell'API scorekeep ha l'API compilata

con Gradle. L'immagine del contenitore del frontend Scorekeep serve il frontend utilizzando il server proxy nginx. Questo server indirizza le richieste ai percorsi che iniziano con /api all'API.

Per analizzare le richieste HTTP in ingresso, l'applicazione aggiunge il `TracingFilter` fornito dall'SDK.

Example src/main/java/scorekeep/WebConfig.java - filtro servlet

```
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
...

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
...
```

Questo filtro invia i dati di tracciamento di tutte le richieste in entrata elaborate dall'applicazione, inclusi URL della richiesta, metodo, stato della risposta, ora di inizio e ora di fine.

L'applicazione effettua anche chiamate downstream a DynamoDB utilizzando. AWS SDK per Java Per strumentare queste chiamate, l'applicazione accetta semplicemente i AWS SDK-related sottomoduli come dipendenze e l' X-Ray SDK per Java strumenta automaticamente tutti i client SDK. AWS

L'applicazione utilizza `Docker` per creare il codice sorgente sull'istanza `Gradle Docker Image` e il `Scorekeep API Dockerfile` file per eseguire il JAR eseguibile che Gradle genera al suo interno. `ENTRYPOINT`

Example utilizzo di Docker per creare tramite Gradle Docker Image

```
docker run --rm -v /PATH/TO/SCOREKEEP_REPO/home/gradle/project -w /home/gradle/project
 gradle:4.3 gradle build
```

Example PUNTO DI INGRESSO del file Dockerfile

```
ENTRYPOINT [ "sh", "-c", "java -Dserver.port=5000 -jar scorekeep-api-1.0.0.jar" ]
```

Il file `build.gradle` attiva lo scaricamento dei sottomoduli SDK da Maven durante la compilazione dichiarandoli come dipendenze.

Example build.gradle -- dipendenze

```
...
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")
    }
}
```

I sottomoduli core, AWS SDK e AWS SDK Instrumentor sono tutto ciò che serve per strumentare automaticamente tutte le chiamate downstream effettuate con l'SDK. AWS

Per inoltrare i dati grezzi del segmento all' X-Ray API, il X-Ray daemon deve ascoltare il traffico sulla porta UDP 2000. A tale scopo, l'applicazione fa eseguire il X-Ray demone in un contenitore che viene distribuito insieme all'applicazione Scorekeep su ECS come contenitore secondario. Per ulteriori informazioni, consulta l'argomento relativo al demone. X-Ray

Example X-Ray Definizione del contenitore daemon in una definizione di attività 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
      ...
```

L' X-Ray SDK per Java fornisce una classe denominata `AWSXRay` che fornisce il registratore globale, `TracingHandler` che puoi usare per strumentare il tuo codice. Puoi configurare la registrazione globale per personalizzare il `AWSXRayServletFilter` che crea i segmenti relativi alle chiamate HTTP in entrata. L'esempio include un blocco statico nella classe `WebConfig` che permette di configurare la registrazione globale con i plug-in e le regole di campionamento.

Example src/main/java/scorekeep/WebConfig.java - registratore

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.plugins.ECSPlugin;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;
...

@Configuration
public class WebConfig {
  ...

  static {
    AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
 ECSPlugin()).withPlugin(new EC2Plugin());

    URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
    builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

    AWSXRay.setGlobalRecorder(builder.build());
    ...

  }
}
```

Questo esempio usa il compilatore per caricare le regole di campionamento da un file denominato `sampling-rules.json`. Le regole di campionamento determinano la frequenza con cui l'SDK registra i segmenti relativi alle richieste in entrata.

Example src/main//sampling-rules.json java/resources

```
{
  "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
  }
}
```

Le regole di campionamento definiscono quattro regole di campionamento personalizzate e la regola di default. Per ogni richiesta in entrata, l'SDK valuta le regole nell'ordine in cui sono definite. L'SDK si applica la prima regola che corrisponde al metodo, al percorso e al nome del servizio della richiesta. Per Scorekeep, la prima regola cattura tutte le richieste POST (chiamate di creazione delle risorse) applicando un obiettivo fisso di una richiesta al secondo e una percentuale di 1,0, o il 100%, delle richieste dopo il raggiungimento dell'obiettivo fisso.

Le altre tre regole personalizzate applicano una percentuale del 5% delle chiamate, senza alcun obiettivo fisso per sessioni, giochi e letture dello stato (richieste GET). Questo riduce al minimo il numero di tracciamenti per le chiamate periodiche che il front-end invoca automaticamente ogni pochi secondi per garantire l'aggiornamento del contenuto. Per tutte le altre richieste, il file definisce un obiettivo di default di una richiesta al secondo e una percentuale del 10% delle chiamate successive.

L'applicazione di esempio mostra anche come utilizzare le funzionalità avanzate come l'analisi manuale del client SDK, la creazione di sottosegmenti aggiuntivi e le chiamate HTTP in uscita. Per ulteriori informazioni, consulta AWS X-Ray applicazione di esempio.

Opzionale: policy con privilegio minimo

I contenitori Scorekeep ECS accedono alle risorse utilizzando politiche di accesso completo, come e. `AmazonSNSFullAccess` `AmazonDynamoDBFullAccess` L'utilizzo di politiche di accesso completo non è la migliore pratica per le applicazioni di produzione. L'esempio seguente aggiorna la politica IAM di DynamoDB per migliorare la sicurezza dell'applicazione. Per saperne di più sulle best practice di sicurezza nelle policy IAM, vedi Identity and access management for. AWS X-Ray

Example definizione del modello cf-resources.yaml ECSTaskRole

```
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"
```

Per aggiornare la tua policy, devi prima identificare l'ARN delle tue risorse DynamoDB. Quindi si utilizza l'ARN in una policy IAM personalizzata. Infine, applichi tale policy al profilo della tua istanza.

Per identificare l'ARN della tua risorsa DynamoDB:

1. Aprire la console DynamoDB.
2. Scegli Tabelle dalla barra di navigazione a sinistra.
3. Scegliete una delle opzioni `scorekeep-*` per visualizzare la pagina dei dettagli della tabella.
4. Nella scheda Panoramica, scegli Informazioni aggiuntive per espandere la sezione e visualizzare Amazon Resource Name (ARN). Copia questo valore.
5. Inserisci l'ARN nella seguente policy IAM, sostituendo `AWS_ACCOUNT_ID` i valori `AWS_REGION` e con la regione e l'ID dell'account specifici. Questa nuova policy consente solo le azioni specificate, invece della `AmazonDynamoDBFullAccess` policy che consente qualsiasi azione.

   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-*"
        }
    ]
}
```

Le tabelle create dall'applicazione seguono una convenzione di denominazione coerente. È possibile utilizzare il `scorekeep-*` formato per indicare tutte le tabelle Scorekeep.

Cambia la tua politica IAM

1. Apri il ruolo dell'attività Scorekeep (ScorekeepRole) dalla console IAM.
2. Seleziona la casella di controllo accanto alla `AmazonDynamoDBFullAccess` policy e scegli Rimuovi per rimuovere questa policy.
3. Scegli Aggiungi autorizzazioni, quindi Allega politiche e infine Crea policy.
4. Scegli la scheda JSON e incolla la policy creata sopra.
5. Scegli Avanti: Tag nella parte inferiore della pagina.
6. Scegli Avanti: Rivedi nella parte inferiore della pagina.
7. In Nome, assegna un nome alla policy.
8. Scegli Crea policy nella parte inferiore della pagina.
9. Allega la policy appena creata al `scorekeepRole` ruolo. Potrebbero essere necessari alcuni minuti prima che la politica allegata abbia effetto.

Se hai allegato il nuovo criterio al `scorekeepRole` ruolo, devi scollegarlo prima di eliminare lo CloudFormation stack, poiché questo criterio allegato bloccherà l'eliminazione dello stack. La policy può essere rimossa automaticamente eliminando la policy.

Rimuovi la tua policy IAM personalizzata

1. Apri la console IAM.
2. Scegli Policy dalla barra di navigazione a sinistra.
3. Cerca il nome della politica personalizzata che hai creato in precedenza in questa sezione e scegli il pulsante di opzione accanto al nome della politica per evidenziarlo.

4. Scegli il menu a discesa Azioni, quindi scegli Elimina.
5. Digita il nome della politica personalizzata, quindi scegli Elimina per confermare l'eliminazione. Questo scollegherà automaticamente la policy dal `scorekeepRole` ruolo.

Eliminazione

Segui questi passaggi per eliminare le risorse dell'applicazione Scorekeep:

> **Note**
>
> Se hai creato e allegato policy personalizzate utilizzando la sezione precedente di questo tutorial, devi rimuovere la policy dalla `scorekeepRole` prima di eliminare lo stack. CloudFormation

Console di gestione AWS

Elimina l'applicazione di esempio utilizzando Console di gestione AWS

1. Apri la console CloudFormation
2. Scegli il pulsante di opzione accanto al nome `scorekeep` dello stack nell'elenco, quindi scegli Elimina.
3. La CloudFormation pila viene ora eliminata. Lo stato dello stack durerà `DELETE_IN_PROGRESS` alcuni minuti fino all'eliminazione di tutte le risorse. Lo stato verrà aggiornato periodicamente oppure puoi aggiornare la pagina.

AWS CLI

Eliminare l'applicazione di esempio utilizzando AWS CLI

1. Immettete il seguente AWS CLI comando per eliminare lo CloudFormation stack:

```
aws cloudformation delete-stack --stack-name scorekeep
```

2. Attendi che lo CloudFormation stack non esista più, operazione che richiederà circa cinque minuti. Usa il AWS CLI comando seguente per controllare lo stato:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

Fasi successive

Scopri di più X-Ray nel prossimo capitolo,AWS X-Ray concetti.

Per personalizzare la tua app, scopri di più sull' X-Ray SDK per Java o su uno degli altri X-Ray SDK:

- X-Ray SDK per Java — AWS X-Ray SDK per Java
- X-Ray SDK per — Node.js AWS X-Ray SDK per Node.js
- X-Ray SDK per .NET — AWS X-Ray SDK per.NET

Per eseguire il X-Ray demone localmente o acceso AWS, vedi. AWS X-Ray demone

Per contribuire all'applicazione di esempio su GitHub, vedere https://github.com/awslabs/eb-java-scorekeep/tree/xray-gettingstarted eb-java-scorekeep.

Strumentazione manuale dei client SDK AWS

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L'X-Ray SDK for Java strumenta automaticamente tutti i client SDK quando includi il AWS sottomodulo SDK Instrumentor nelle AWS dipendenze di compilazione.

Puoi disabilitare l'analisi automatica del client rimuovendo il sottomodulo Instrumentor. In questo modo puoi analizzare manualmente alcuni client escludendone altri, oppure utilizzare diversi gestori del tracciamento su diversi client.

Per illustrare il supporto per la strumentazione di client AWS SDK specifici, l'applicazione passa un gestore di tracciamento come gestore di richieste nel modello utente, di gioco e di `AmazonDynamoDBClientBuilder` sessione. Questa modifica al codice indica all'SDK di strumentare tutte le chiamate a DynamoDB utilizzando tali client.

Example **`src/main/java/scorekeep/SessionModel.java`**— Strumentazione manuale del client SDK 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 rimuovi il sottomodulo AWS SDK Instrumentor dalle dipendenze del progetto, nella mappa di traccia vengono visualizzati solo i client SDK con strumentazione AWS manuale.

Creazione di sottosegmenti aggiuntivi

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Nella classe del modello utente, l'applicazione crea manualmente dei sottosegmenti per raggruppare tutte le chiamate a valle effettuate all'interno della funzione `saveUser` e aggiunge i metadati.

Example **`src/main/java/scorekeep/UserModel.java`** - Sottosegmenti personalizzati

```
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();
    }
  }
```

Registrazione di annotazioni, metadati e ID utente

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

Nella classe Game Model, l'applicazione registra `Game` gli oggetti in un blocco di metadati ogni volta che salva una partita in DynamoDB. Separatamente, l'applicazione memorizza gli ID delle partite nelle annotazioni per l'uso con espressioni filtro.

Example **`src/main/java/scorekeep/GameModel.java`** — Annotazioni e metadati

```
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();
    }
  }
```

Nel controllo delle mosse, l'applicazione memorizza gli ID utente con `setUser`. Gli ID utente vengono memorizzati in un campo separato su segmenti e sono indicizzati per l'uso nelle ricerche.

Example src/main/java/scorekeep/MoveController.java — ID utente

```
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);
  }
```

Analisi delle chiamate HTTP in uscita

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione — Il 25 febbraio 2026, AWS X-Ray SDKs/Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica

dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

La classe user factory mostra come l'applicazione utilizza l' X-Ray SDK per la versione di Java per strumentare le chiamate HTTP in uscita. `HTTPClientBuilder`

Example **`src/main/java/scorekeep/UserFactory.java`** — Strumentazione HttpClient

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;

  public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
    HttpGet httpGet = new HttpGet("http://uinames.com/api/");
    CloseableHttpResponse response = httpclient.execute(httpGet);
    try {
      HttpEntity entity = response.getEntity();
      InputStream inputStream = entity.getContent();
      ObjectMapper mapper = new ObjectMapper();
      Map<String, String> jsonMap = mapper.readValue(inputStream, Map.class);
      String name = jsonMap.get("name");
      EntityUtils.consume(entity);
      return name;
    } finally {
      response.close();
    }
  }
```

Se al momento utilizzi `org.apache.http.impl.client.HttpClientBuilder`, è possibile sostituire l'istruzione di importazione della classe con una che includa `com.amazonaws.xray.proxies.apache.http.HttpClientBuilder`.

Analizzare le chiamate a un database PostgresSQL

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del

supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

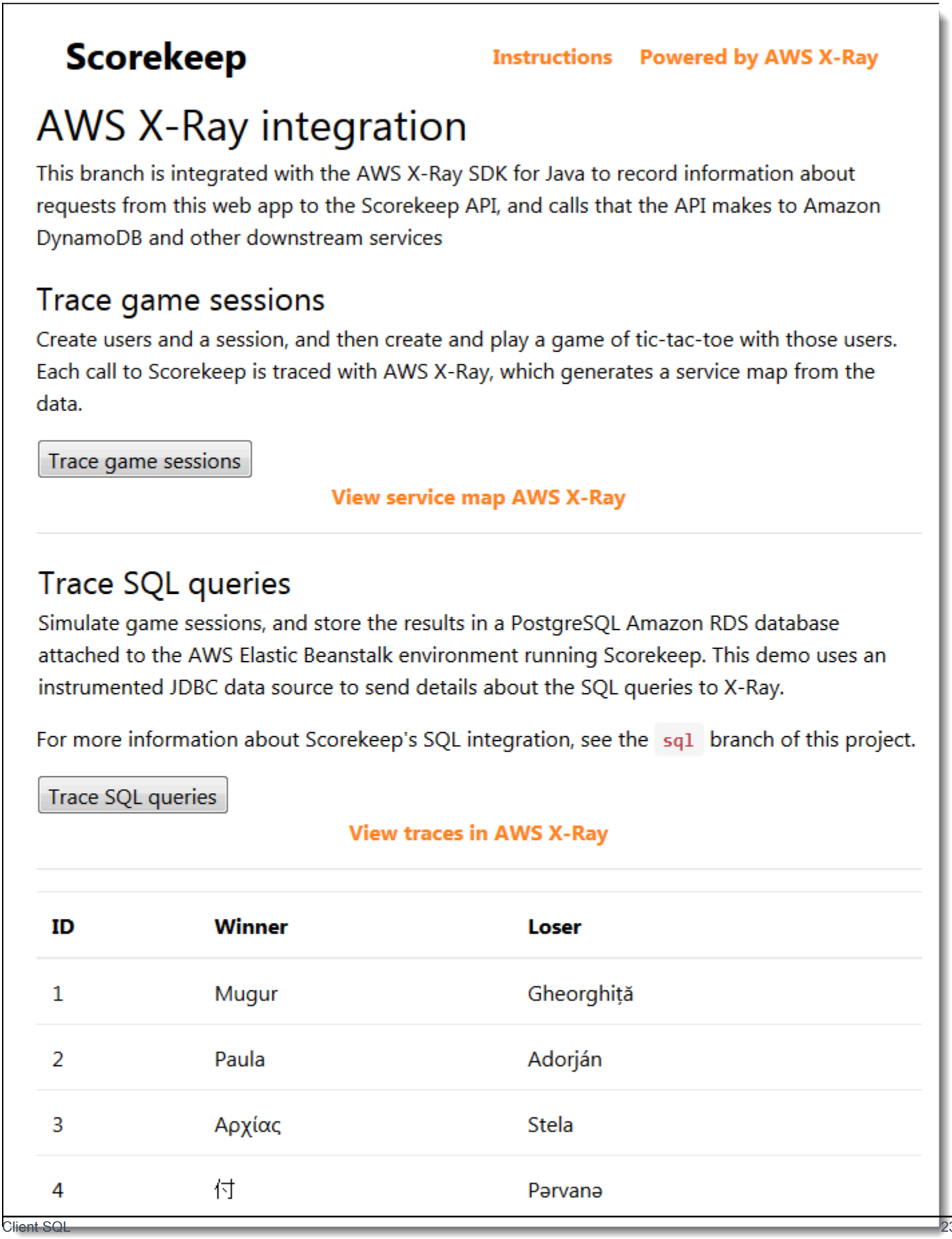
Il `application-pgsql.properties` file aggiunge l' X-Ray intercettore di tracciamento PostgreSQL all'origine dati creata in. `RdsWebConfig.java`

Example **`application-pgsql.properties`** — Strumentazione del database 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
```

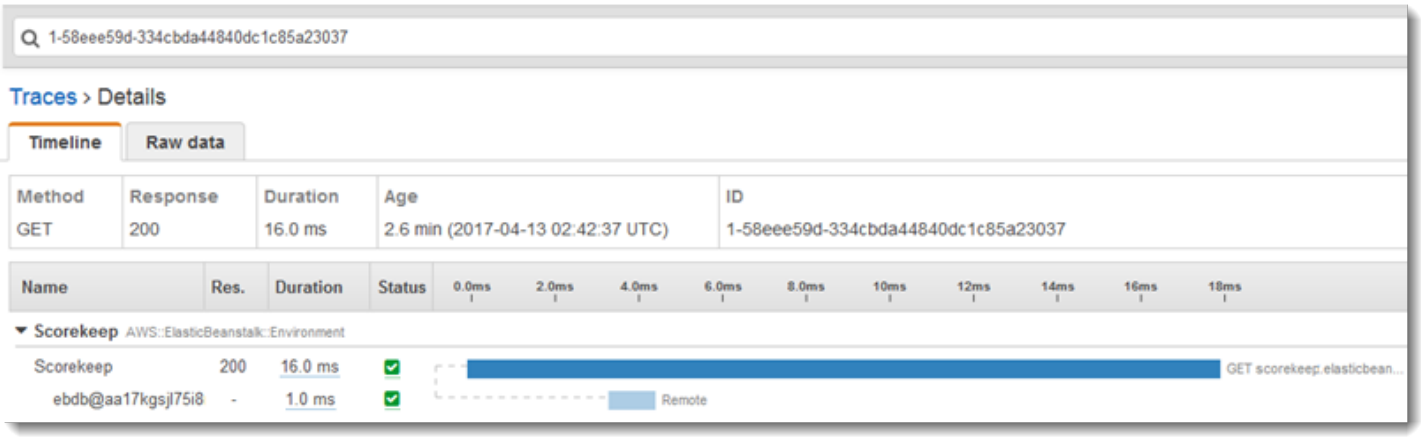
> **Note**
>
> Per ulteriori informazioni su come aggiungere un database PostgreSQL all'ambiente dell'applicazione, consulta Configurazione dei database con Elastic Beanstalk nella Guida per lo sviluppatore di AWS Elastic Beanstalk .

La pagina X-Ray dimostrativa della `xray` filiale include una demo che utilizza la fonte di dati strumentata per generare tracce che mostrano informazioni sulle query SQL generate. Vai al `/#/xray` percorso nell'applicazione in esecuzione o scegli Powered by AWS X-Ray nella barra di navigazione per visualizzare la pagina demo.


Scorekeep
Instructions
Powered by AWS X-Ray
AWS X-Ray integration
This branch is integrated with the AWS X-Ray SDK for Java to record information about requests from this web app to the Scorekeep API, and calls that the API makes to Amazon DynamoDB and other downstream services
Trace game sessions
Create users and a session, and then create and play a game of tic-tac-toe with those users. Each call to Scorekeep is traced with AWS X-Ray, which generates a service map from the data.
Trace game sessions
View service map AWS X-Ray
Trace SQL queries
Simulate game sessions, and store the results in a PostgreSQL Amazon RDS database attached to the AWS Elastic Beanstalk environment running Scorekeep. This demo uses an instrumented JDBC data source to send details about the SQL queries to X-Ray.
For more information about Scorekeep's SQL integration, see the sql branch of this project.
Trace SQL queries
View traces in AWS X-Ray
ID
Winner
Loser
1
Mugur
Gheorghiță
2
Paula
Adorján
3
Αρχίας
Stela
4
付
Pərvanə

Scegli Trace SQL queries (Traccia query SQL) per simulare le sessioni di gioco e memorizzare i risultati nel database collegato. Quindi, scegli Visualizza tracce in AWS X-Ray per visualizzare un elenco filtrato di tracce che raggiungono il percorso dell'API. `/api/history`

Scegli uno dei tracciamenti nell'elenco per visualizzare la sequenza temporale, inclusa la query SQL.



• Strumentazione AWS Lambda funzioni

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione — Il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

AWS Lambda Scorekeep utilizza due funzioni. La prima è una Node.js funzione del `lambda` ramo che genera nomi casuali per i nuovi utenti. Quando un utente crea una sessione senza immettere un nome, l'applicazione chiama una funzione denominata `random-name` con il AWS SDK per Java. L' X-Ray SDK per Java registra le informazioni sulla chiamata a Lambda in un sottosegmento come qualsiasi altra chiamata effettuata con un client SDK AWS strumentato.

> **Note**
>
> L'esecuzione della funzione `random-name` Lambda richiede la creazione di risorse aggiuntive al di fuori dell'ambiente Elastic Beanstalk. Per ulteriori informazioni e istruzioni, consulta il file readme: Lambda Integration. AWS

La seconda funzione, `scorekeep-worker`, è una funzione Python che viene eseguita indipendentemente dall'API Scorekeep. Quando una partita termina, l'API scrive l'ID della sessione e l'ID della partita su una coda SQS. La funzione worker legge gli elementi dalla coda e chiama l'API Scorekeep per creare record completi di ogni sessione di gioco per l'archiviazione in Amazon S3.

Scorekeep include CloudFormation modelli e script per creare entrambe le funzioni. Poiché è necessario raggruppare l' X-Ray SDK con il codice della funzione, i modelli creano le funzioni senza alcun codice. Quando distribuisci Scorekeep, un file di configurazione incluso nella cartella `.ebextensions` crea un pacchetto di codice sorgente che include l'SDK e aggiorna il codice della funzione e la configurazione con AWS Command Line Interface.

Funzioni

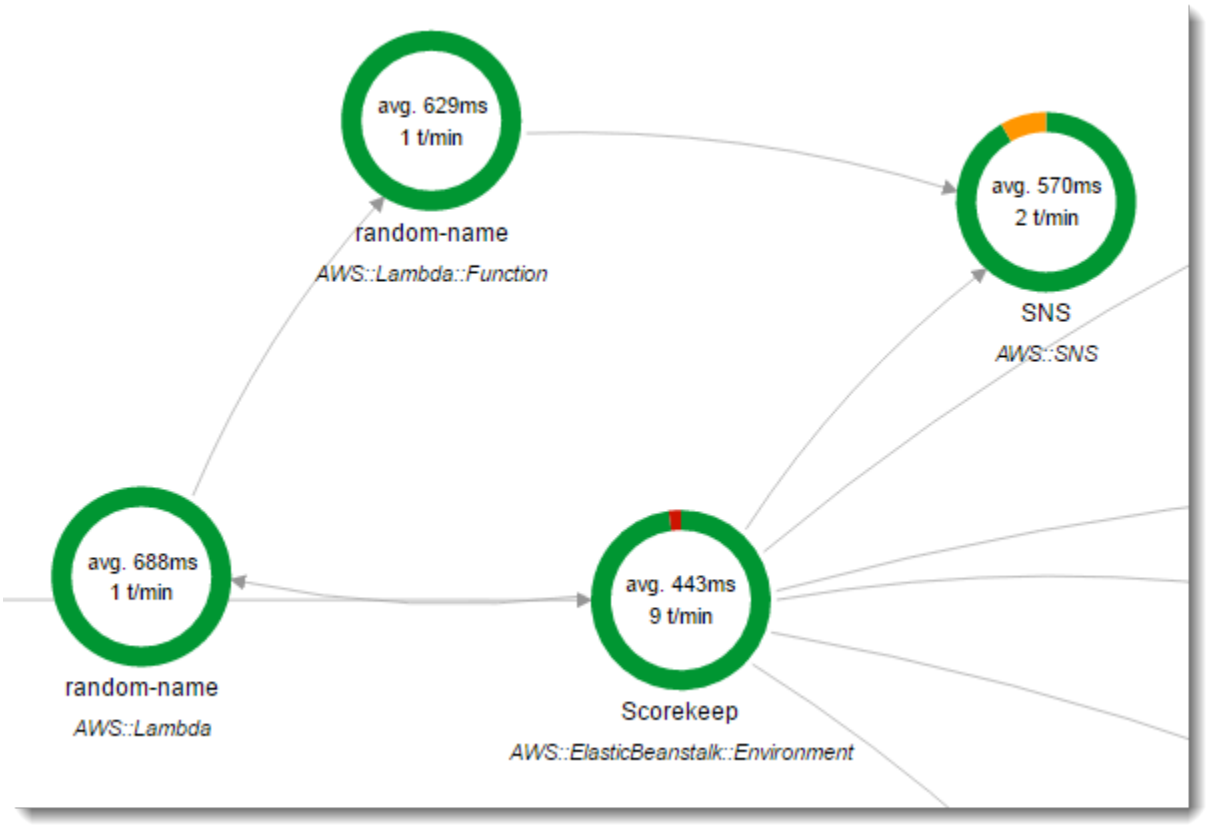
- Nome casuale
- Worker

Nome casuale

Scorekeep richiama la funzione di generazione del nome casuale quando un utente avvia una sessione di gioco senza aver eseguito l'accesso oppure senza aver specificato un nome utente. Quando Lambda elabora la chiamata `arandom-name`, legge l'intestazione di tracciamento, che contiene l'ID di traccia e la decisione di campionamento scritti dall'SDK per Java. X-Ray

Per ogni richiesta campionata, Lambda esegue il demone e scrive due segmenti. X-Ray Il primo segmento registra le informazioni sulla chiamata a Lambda che richiama la funzione. Questo segmento contiene le stesse informazioni del sottosegmento registrato da Scorekeep, ma dal punto di vista Lambda. Il secondo segmento rappresenta l'attività svolta dalla funzione.

Lambda passa il segmento della funzione all' X-Ray SDK tramite il contesto della funzione. Quando si utilizza una funzione Lambda, non si utilizza l'SDK per creare un segmento per le richieste in arrivo. Lambda fornisce il segmento e tu usi l'SDK per strumentare i client e scrivere sottosegmenti.



La `random-name` funzione è implementata in. Node.js Utilizza l'SDK for JavaScript in per Node.js inviare notifiche con Amazon SNS e l' X-Ray SDK per Node.js strumentare il client SDK. AWS Per scrivere annotazioni, la funzione crea una sottosegmento personalizzato con `AWSXRay.captureFunc` e scrive le annotazioni nella funzione analizzata. In Lambda, non puoi scrivere annotazioni direttamente nel segmento di funzione, ma solo in un sottosegmento creato da te.

Example **`function/index.js`** -- Funzione Lambda con nome casuale

```
var AWSXRay = require('aws-xray-sdk-core');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));

AWS.config.update({region: process.env.AWS_REGION});
var Chance = require('chance');

var myFunction = function(event, context, callback) {
  var sns = new AWS.SNS();
  var chance = new Chance();
  var userid = event.userid;
  var name = chance.first();
```

```
  AWSXRay.captureFunc('annotations', function(subsegment){
    subsegment.addAnnotation('Name', name);
    subsegment.addAnnotation('UserID', event.userid);
  });

  // Notify
  var params = {
    Message: 'Created randon name "' + name + '"" for user "' + userid + '".',
    Subject: 'New user: ' + name,
    TopicArn: process.env.TOPIC_ARN
  };
  sns.publish(params, function(err, data) {
    if (err) {
      console.log(err, err.stack);
      callback(err);
    }
    else {
      console.log(data);
      callback(null, {"name": name});
    }
  });
};

exports.handler = myFunction;
```

Questa funzione viene creata automaticamente quando distribuisci l'applicazione di esempio su Elastic Beanstalk. Il `xray` ramo include uno script per creare una funzione Lambda vuota. I file di configurazione presenti nella `.ebextensions` cartella generano il pacchetto di funzioni npm `install` durante la distribuzione, quindi aggiornano la funzione Lambda con la CLI. AWS

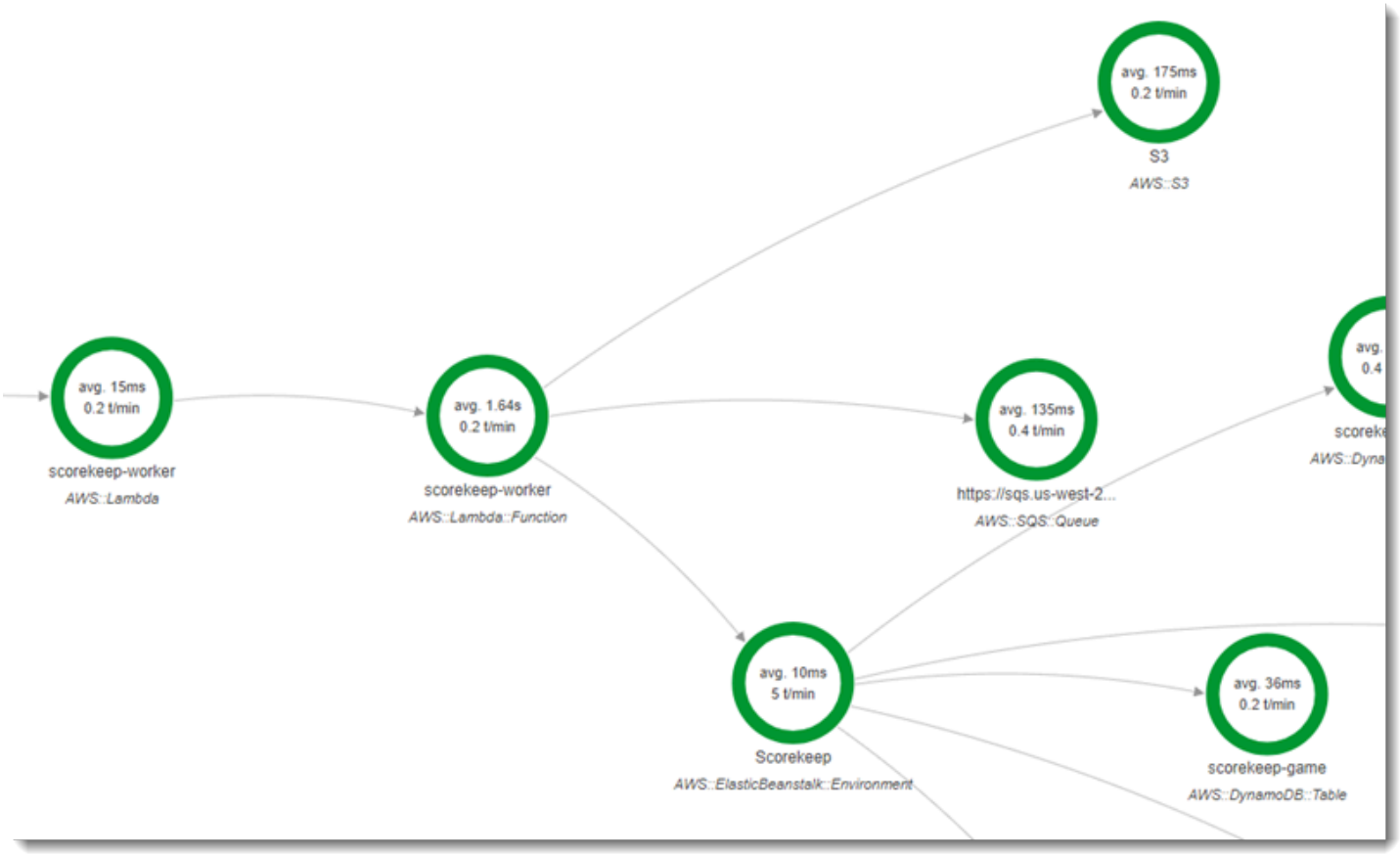
Worker

La funzione worker analizzata viene fornito nel suo ramo, `xray-worker`, poiché non può essere eseguita a meno che non si creino prima la funzione worker e le relative risorse. Per le istruzioni, consulta il readme del ramo.

La funzione viene attivata da un CloudWatch evento Amazon Events in bundle ogni 5 minuti. Quando viene eseguita, la funzione estrae un elemento da una coda Amazon SQS gestita da Scorekeep. Ogni messaggio contiene informazioni su una partita completata.

Il worker preleva il record della partita e i documenti dalle altre tabelle a cui il record della partita fa riferimento. Ad esempio, il record di gioco in DynamoDB include un elenco di mosse eseguite durante la partita. L'elenco non contiene le mosse, ma gli ID delle mosse che sono memorizzate in una tabella separata.

Anche le sessioni e gli stati sono memorizzati come riferimenti. In questo modo si evita che le voci nella tabella delle partite diventino troppo grandi ma sono necessarie chiamate aggiuntive per recuperare tutte le informazioni sulla partita. Il lavoratore dereferenzia tutte queste voci e crea un record completo del gioco come un unico documento in Amazon S3. Quando desideri eseguire analisi sui dati, puoi eseguire query su di essi direttamente in Amazon S3 con Amazon Athena senza eseguire migrazioni di dati ad alto volume di lettura per estrarli da DynamoDB.



La funzione worker dispone del tracciamento attivo abilitato nella sua configurazione in AWS Lambda. A differenza della funzione di nome casuale, il lavoratore non riceve una richiesta da un'applicazione strumentata, quindi AWS Lambda non riceve un'intestazione di tracciamento. Con il tracciamento attivo, Lambda crea l'ID di traccia e prende decisioni di campionamento.

L' X-Ray SDK per Python è composto da poche righe nella parte superiore della funzione che importa l'SDK ed esegue la sua `patch_all` funzione per applicare patch ai client HTTP e ai client HTTP

utilizzati per chiamare Amazon SQS AWS SDK per Python (Boto) e Amazon S3. Quando il worker richiama l'API Scorekeep, l'SDK aggiunge l'intestazione di tracciamento alla richiesta per tracciare le chiamate tramite l'API.

Example_lambda/scorekeep- -worker.py worker/scorekeep -- Funzione Worker Lambda

```
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')

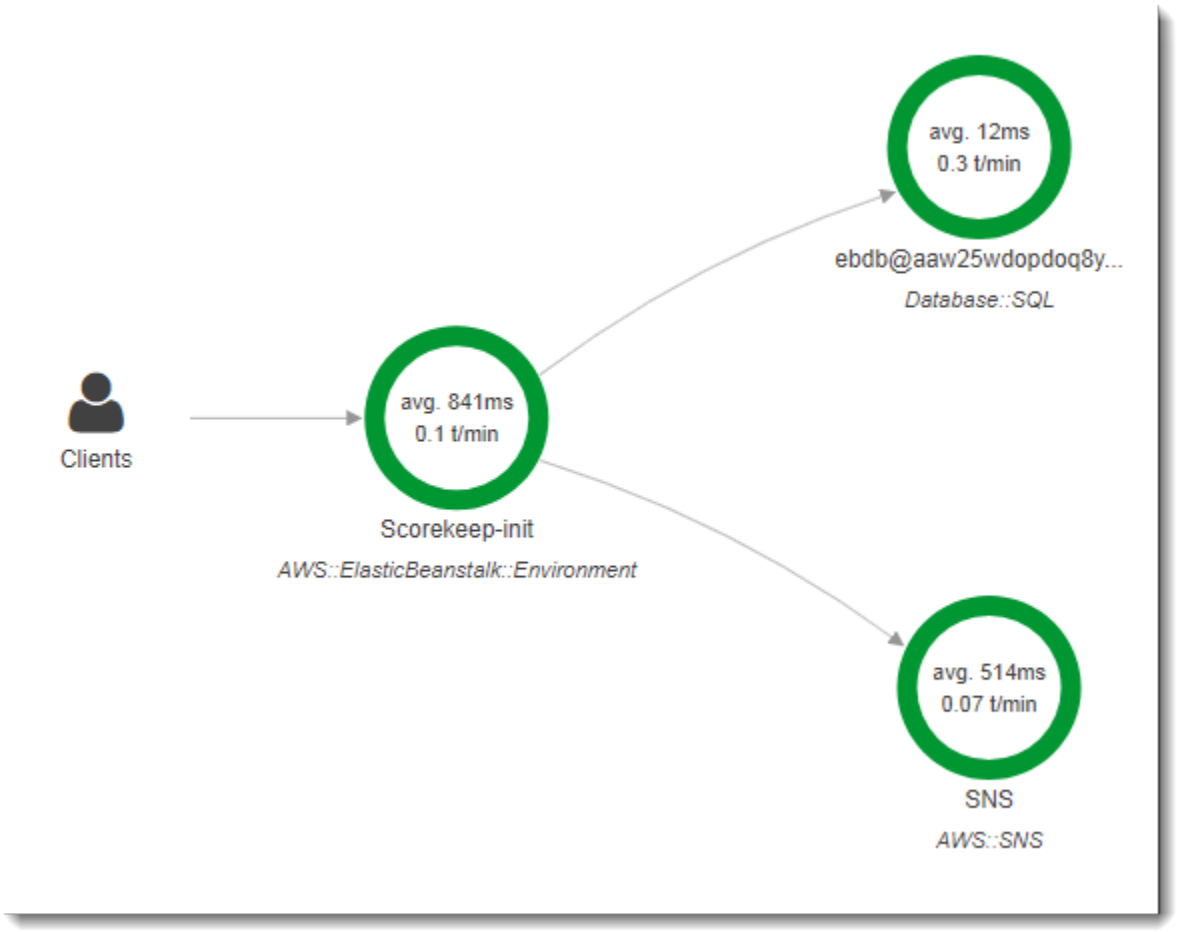
    # Receive message from SQS queue
    response = sqs.receive_message(
        QueueUrl=queue_url,
        AttributeNames=[
            'SentTimestamp'
        ],
        MaxNumberOfMessages=1,
        MessageAttributeNames=[
            'All'
        ],
        VisibilityTimeout=0,
        WaitTimeSeconds=0
    )
   ...
```

Analisi del codice di avvio

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

L' X-Ray SDK for Java crea automaticamente segmenti per le richieste in arrivo. Se una richiesta rientra nell'ambito, puoi utilizzare i client analizzati e registrare i sottosegmenti senza problemi. Se provi a utilizzare un client con strumenti nel codice di avvio, tuttavia, otterrai un. SegmentNotFoundException

Il codice di avvio viene eseguito al di fuori del request/response flusso standard di un'applicazione Web, quindi è necessario creare segmenti manualmente per strumentarlo. Scorekeep mostra l'analisi del codice di avvio nel suo file `WebConfig`. Scorekeep chiama un database SQL e Amazon SNS durante l'avvio.



La `WebConfig` classe predefinita crea un abbonamento Amazon SNS per le notifiche. Per fornire un segmento su cui scrivere all' X-Ray SDK quando viene utilizzato il client Amazon SNS, le `beginSegment` chiamate Scorekeep `endSegment` e sul registratore globale.

Example **src/main/java/scorekeep/WebConfig.java** — Strumentato AWS Client SDK nel codice di avvio

```
AWSXRay.beginSegment("Scorekeep-init");
if ( System.getenv("NOTIFICATION_EMAIL") != null ){
  try { Sns.createSubscription(); }
  catch (Exception e ) {
    logger.warn("Failed to create subscription for email "+
 System.getenv("NOTIFICATION_EMAIL"));
  }
}
AWSXRay.endSegment();
```

In`RdsWebConfig`, che Scorekeep utilizza quando è connesso un database Amazon RDS, la configurazione crea anche un segmento per il client SQL che Hibernate utilizza quando applica lo schema del database durante l'avvio.

Example **`src/main/java/scorekeep/RdsWebConfig.java`** — Client di database SQL strumentato nel codice di avvio

```
@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` viene eseguito automaticamente e utilizza un client SQL. Poiché il client è analizzato, Scorekeep deve sostituire l'impostazione predefinita e fornire un segmento che l'SDK possa utilizzare quando il client viene richiamato.

Analisi degli script

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare

a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

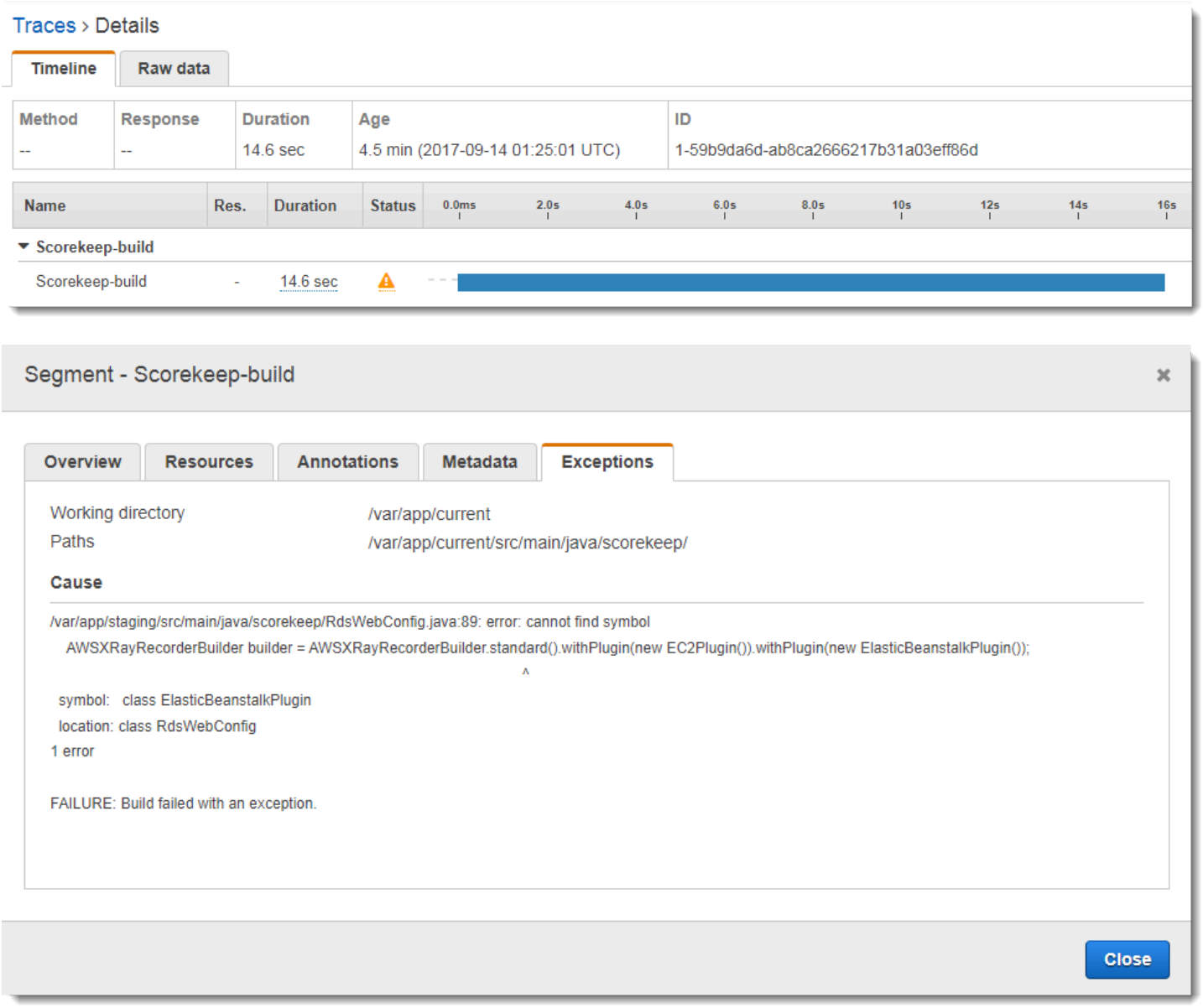
Puoi anche analizzare del codice che non fa parte della tua applicazione. Quando il X-Ray demone è in esecuzione, inoltrerà tutti i segmenti che riceve, anche se non sono generati dall' X-RaySDK. X-Ray Scorekeep usa i propri script per analizzare il processo di build che compila l'applicazione durante la distribuzione.

Example **bin/build.sh** — Script di compilazione con strumenti

```
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 e xray_success.py sono semplici script Python che costruiscono oggetti segmenti, li convertono in documenti in formato JSON e li inviano al daemon su UDP. Se la build di Gradle fallisce, puoi trovare il messaggio di errore facendo clic sul nodo scorekeep-build nella mappa di traccia della console. X-Ray





Analisi di un client di app Web

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare

a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Nella `xray-cognito`filiale, Scorekeep utilizza Amazon Cognito per consentire agli utenti di creare un account e di accedere con esso per recuperare le informazioni utente da un pool di utenti Amazon Cognito. Quando un utente accede, Scorekeep utilizza un pool di identità Amazon Cognito per ottenere credenziali AWS temporanee da utilizzare con. AWS SDK per JavaScript

Il pool di identità viene configurato per garantire agli utenti che accedono l'accesso in scrittura ai dati di tracciamento verso AWS X-Ray. L'applicazione web utilizza queste credenziali per registrare l'ID utente dell'utente cha ha eseguito l'accesso, il percorso del browser e la vista del client delle chiamate all'API di Scorekeep.

La maggior parte del lavoro è effettuato in una classe denominata `xray`. Questa classe di servizio fornisce metodi per generare gli identificatori richiesti, creare segmenti in corso, finalizzare segmenti e inviare documenti di segmento all'API X-Ray.

Example **`public/xray.js`**— Registrazione e caricamento di segmenti

```
...
  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);
      }
    })
  }
```

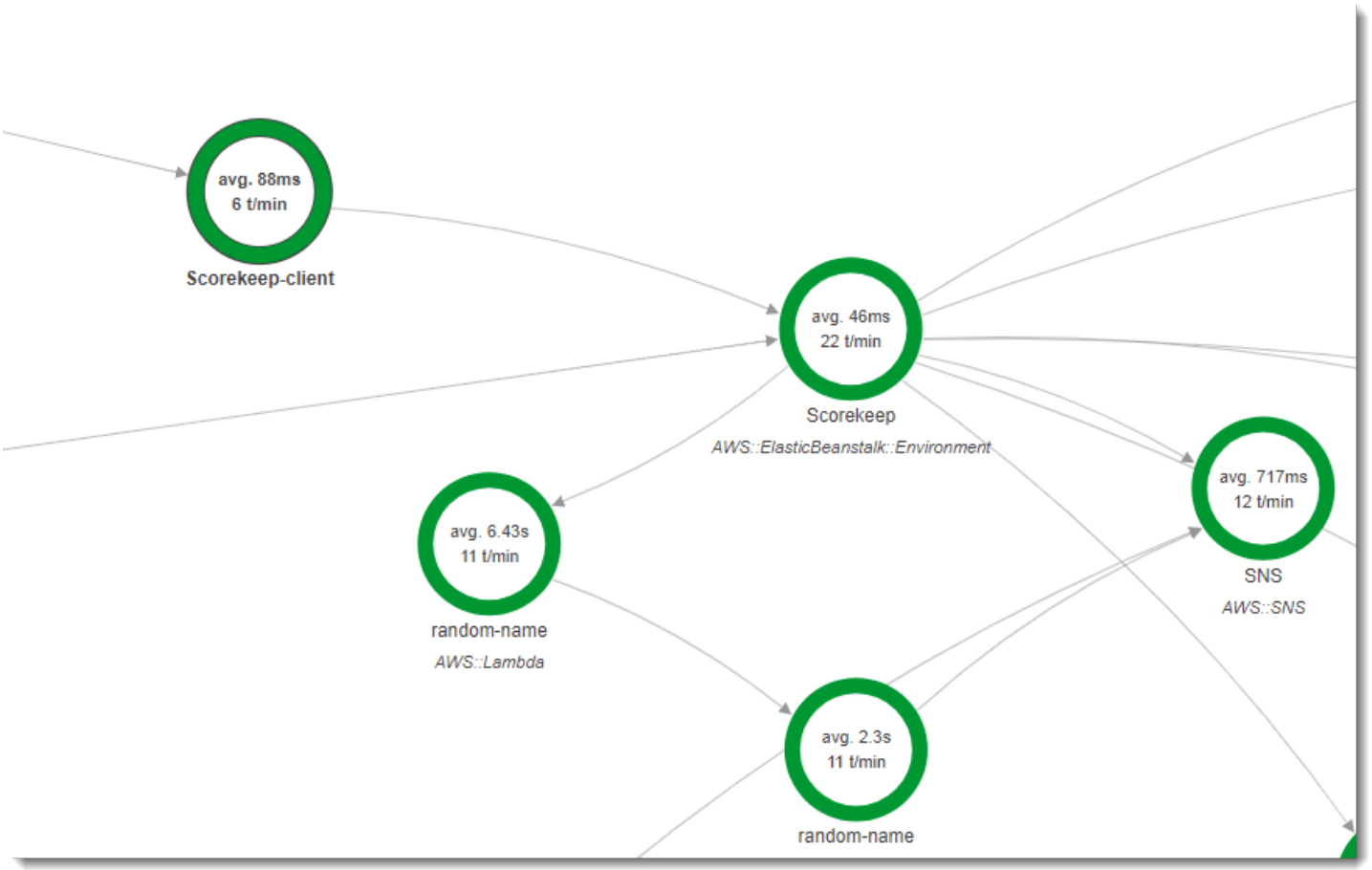
Questi metodi sono chiamati nell'intestazione e nelle funzioni `transformResponse` nei servizi della risorsa che l'app web Scorekeep utilizza per chiamare l'API. Per includere il segmento client nella stessa traccia del segmento generato dall'API, l'app Web deve includere l'ID di traccia e l'ID del segmento in un'intestazione di tracciamento (`X-Amzn-Trace-Id`) leggibile dall'SDK X-Ray. Quando l'applicazione Java con strumentazione riceve una richiesta con questa intestazione, X-Ray SDK for Java utilizza lo stesso ID di traccia e rende il segmento del client dell'app Web il principale del segmento.

Example **`public/app/services.js`**— Registrazione di segmenti per chiamate di risorse angolari e scrittura di intestazioni di tracciamento

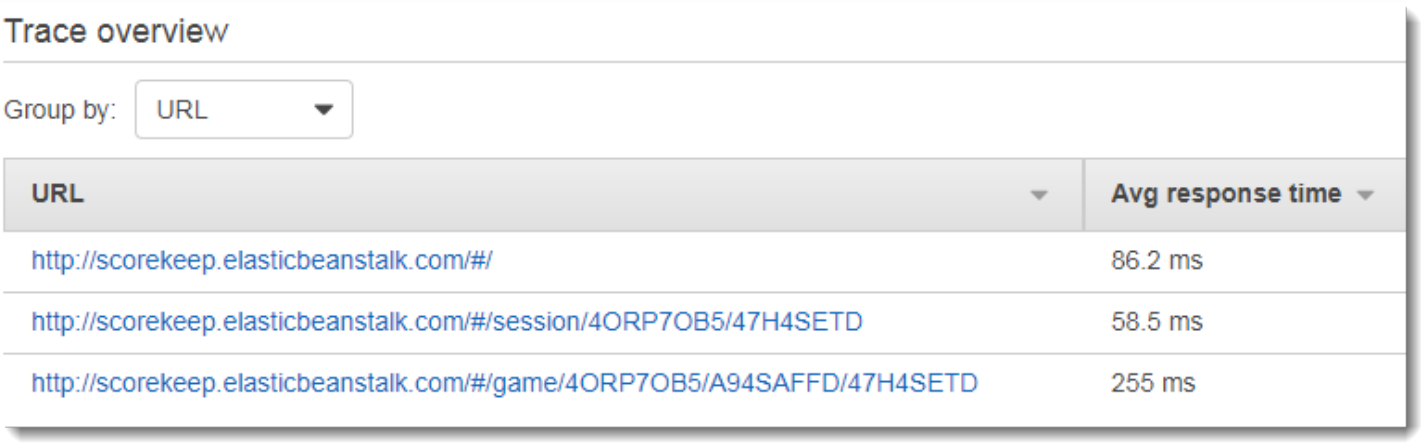
```
var module = angular.module('scorekeep');
module.factory('SessionService', function($resource, api, XRay) {
  return $resource(api + 'session/:id', { id: '@_id' }, {
    segment: {},
```

```
    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);
      },
    },
...
```

La mappa di traccia risultante include un nodo per il client dell'app Web.



I tracciamenti che includono segmenti dall'app web mostrano l'URL che l'utente visualizza all'interno del browser (percorso che inizia con `/#/`). Senza l'analisi del client, ricevi solo l'URL dell'API della risorsa che l'app web chiama (percorsi che iniziano con `/api/`).



Utilizzo dei client analizzati nei thread worker

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Scorekeep utilizza un thread di lavoro per pubblicare una notifica su Amazon SNS quando un utente vince una partita. La pubblicazione della notifica richiede più tempo del resto delle operazioni di richiesta sommate e non ha effetto sul client o sull'utente. Pertanto, eseguire l'attività in modo asincrono è un buon modo per migliorare i tempi di risposta.

Tuttavia, l'X-Ray SDK for Java non sa quale segmento era attivo al momento della creazione del thread. Di conseguenza, quando si tenta di utilizzare il AWS SDK per Java client strumentato all'interno del thread, viene generato un `SegmentNotFoundException` messaggio che blocca il thread.

Example Web-1.error.log

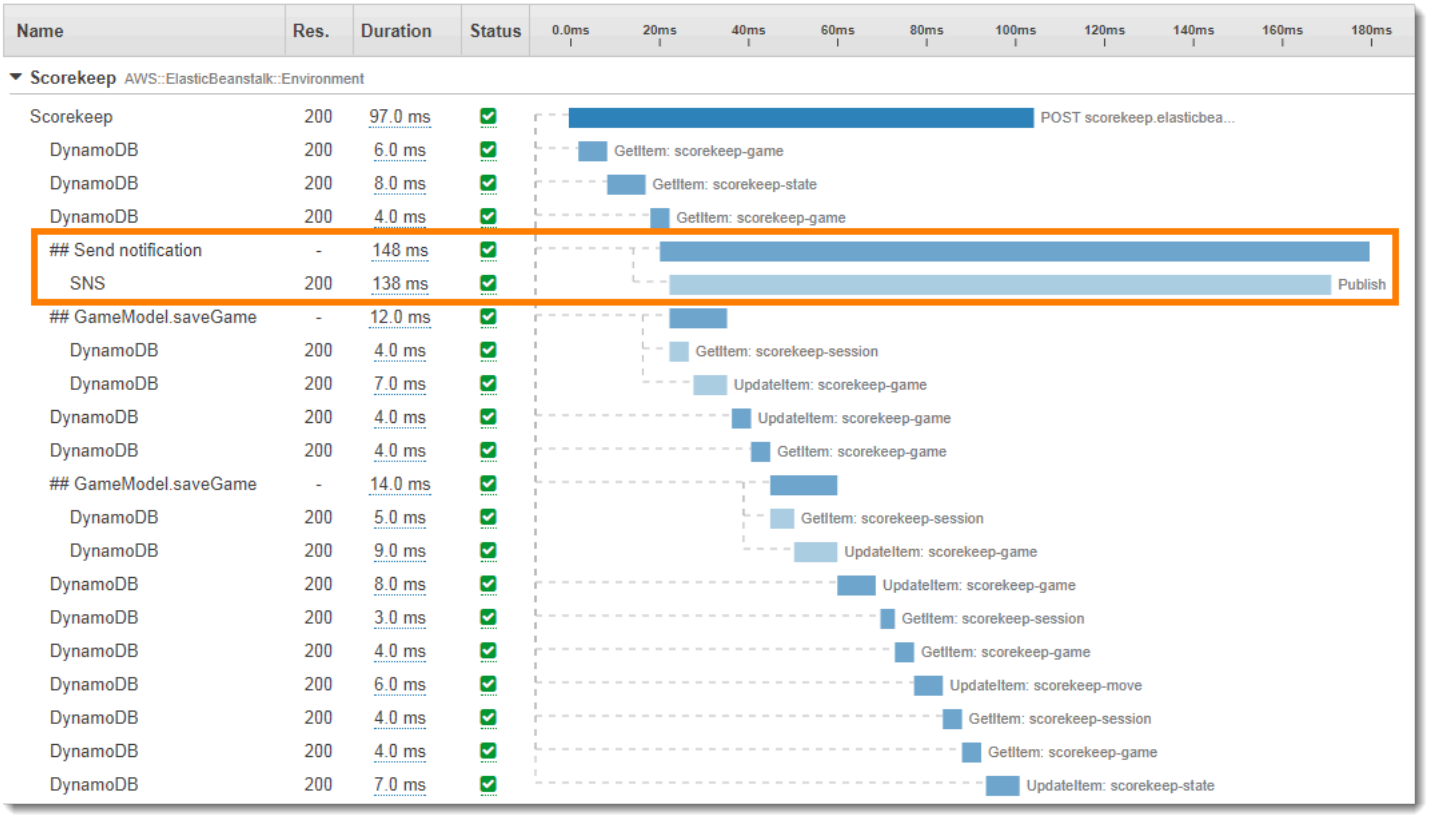
```
Exception in thread "Thread-2" com.amazonaws.xray.exceptions.SegmentNotFoundException:
 Failed to begin subsegment named 'AmazonSNS': segment cannot be found.
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at
 sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at
 sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
...
```

Per risolvere questo problema, l'applicazione utilizza la funzione di `GetTraceEntity` ottenere un riferimento al segmento nel thread principale e `Entity.run()` di eseguire in sicurezza il codice del thread di lavoro con accesso al contesto del segmento.

Example **src/main/java/scorekeep/MoveFactory.java**— Passaggio del contesto di traccia a un thread di lavoro

```
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();
          }
        }
```

Poiché la richiesta è ora risolta prima della chiamata ad Amazon SNS, l'applicazione crea un sottosegmento separato per il thread. Ciò impedisce all'SDK X-Ray di chiudere il segmento prima di registrare la risposta da Amazon SNS. Se nessun sottosegmento è aperto quando Scorekeep ha risolto la richiesta, la risposta di Amazon SNS potrebbe andare persa.



Per ulteriori informazioni sul multithreading, consulta Passaggio del contesto del segmento tra thread in un'applicazione multithread.

AWS X-Ray demone

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

> **Note**
>
> Ora puoi utilizzare l' CloudWatch agente per raccogliere parametri, log e tracce dalle istanze Amazon EC2 e dai server locali. CloudWatch la versione dell'agente 1.300025.0 e successive può raccogliere tracce dal nostro client OpenTelemetryX-Ray SDKs e inviarle a X-Ray. L'utilizzo dell' CloudWatch agente al posto del demone AWS Distro for OpenTelemetry (ADOT) Collector o X-Ray per raccogliere le tracce può aiutare a ridurre il numero di agenti da gestire. Per ulteriori informazioni, consultate l'argomento relativo all'CloudWatch agente nella Guida per l'utente CloudWatch .

Il AWS X-Ray daemon è un'applicazione software che ascolta il traffico sulla porta UDP 2000, raccoglie i dati grezzi dei segmenti e li inoltra all'API. AWS X-Ray Il daemon funziona insieme a AWS X-Ray SDKs e deve essere in esecuzione in modo che i dati inviati da SDKs possano raggiungere il servizio X-Ray. Il demone X-Ray è un progetto open source. Puoi seguire il progetto e inviare problemi e richieste pull su GitHub: github. com/aws/aws-xray-daemon

Su AWS Lambda e AWS Elastic Beanstalk, usa l'integrazione di questi servizi con X-Ray per eseguire il demone. Lambda esegue il demone automaticamente ogni volta che viene richiamata una funzione per una richiesta campionata. Su Elastic Beanstalk, `XRayEnabled` usa l'opzione di configurazione per eseguire il demone sulle istanze del tuo ambiente. Per ulteriori informazioni, consultare la pagina

Per eseguire il demone X-Ray localmente, in locale o in altro modo, è necessario scaricarlo Servizi AWS, eseguirlo e quindi autorizzarlo a caricare i documenti dei segmenti su X-Ray.

Download del daemon

Puoi scaricare il daemon da Amazon S3, Amazon ECR o Docker Hub, quindi eseguirlo localmente o installarlo su un'istanza Amazon EC2 all'avvio.

Amazon S3

Installatori ed eseguibili dei daemon X-Ray

- Linux (eseguibile) — (sig) **aws-xray-daemon-linux-3.x.zip**
- Linux (programma di installazione RPM) — aws-xray-daemon-3.x.rpm
- Linux (programma di installazione DEB) — aws-xray-daemon-3.x.deb
- Linux (ARM64, eseguibile) — **aws-xray-daemon-linux-arm64-3.x.zip**(sig)
- Linux (programma ARM64 di installazione RPM) — aws-xray-daemon-arm64-3.x.rpm
- Linux (programma di ARM64 installazione DEB) — aws-xray-daemon-arm64-3.x.deb
- OS X (eseguibile) — **aws-xray-daemon-macos-3.x.zip**(sig)
- Windows (eseguibile) — aws-xray-daemon-windows-process-3.x.zip(sig)
- Windows (servizio) — aws-xray-daemon-windows-service-3.x.zip(sig)

Questi collegamenti rimandano sempre all'ultima versione 3.x del demone. Per scaricare una versione specifica, procedi come segue:

- Se desideri scaricare una versione precedente alla versione`3.3.0`, sostituiscila `3.x` con il numero di versione. Ad esempio, `2.1.0`. Prima della versione`3.3.0`, l'unica architettura disponibile è`arm64`. Ad esempio `2.1.0` e `arm64`.
- Se desideri scaricare una versione dopo l'altra`3.3.0`, sostituiscila `3.x` con il numero di versione e `arch` con il tipo di architettura.

Le risorse X-Ray vengono replicate su bucket in ogni regione supportata. Per utilizzare il bucket più vicino a te o alle tue AWS risorse, sostituisci la regione nei link precedenti con la tua regione.

```
https://s3.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-daemon/aws-xray-daemon-3.x.rpm
```

Amazon ECR

A partire dalla versione 3.2.0, il daemon è disponibile su Amazon ECR. Prima di estrarre un'immagine, devi autenticare il tuo client docker nel registro pubblico Amazon ECR.

Estrai il tag della versione 3.x rilasciata più recente eseguendo il seguente comando:

```
docker pull public.ecr.aws/xray/aws-xray-daemon:3.x
```

Le versioni precedenti o alpha possono essere scaricate sostituendole `3.x` con `alpha` o con un numero di versione specifico. Non è consigliabile utilizzare un'immagine daemon con un tag alfa in un ambiente di produzione.

Docker Hub

Il demone può essere trovato su Docker Hub. Per scaricare l'ultima versione 3.x rilasciata, esegui il seguente comando:

```
docker pull amazon/aws-xray-daemon:3.x
```

Le versioni precedenti del demone possono essere rilasciate sostituendole `3.x` con la versione desiderata.

Verifica della firma dell'archivio del daemon

I file di firma GPG per le risorse del daemon compresse in archivi ZIP sono inclusi. La chiave pubblica si trova qui: `aws-xray.gpg`.

Puoi utilizzare la chiave pubblica per verificare che l'archivio ZIP del daemon sia originale e non modificato. Prima di tutto, importa la chiave pubblica con GnuPG.

Per importare la chiave pubblica

1. Scarica la chiave pubblica.

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
$ wget $BUCKETURL/xray-daemon/aws-xray.gpg
```

2. Importa la chiave pubblica nel tuo keyring.

```
$ 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
```

Utilizza la chiave importata per verificare la firma del dell'archivio ZIP del daemon.

Per verificare la firma di un archivio

1. Scaricare l'archivio e il file della firma.

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
$ wget $BUCKETURL/xray-daemon/aws-xray-daemon-linux-3.x.zip
$ wget $BUCKETURL/xray-daemon/aws-xray-daemon-linux-3.x.zip.sig
```

2. Eseguire `gpg --verify` per verificare la firma.

```
$ gpg --verify aws-xray-daemon-linux-3.x.zip.sig aws-xray-daemon-linux-3.x.zip
gpg: Signature made Wed 19 Apr 2017 05:06:31 AM UTC using RSA key ID FE6157D3
gpg: Good signature from "AWS X-Ray <aws-xray@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EA6D 9271 FBF3 6990 277F  4B87 7BFE 036B FE61 57D3
```

Prendi nota dell'avviso sulla trust. Una chiave è considerata attendibile solo se è stata firmata da te o da un utente attendibile. Questo non significa che la firma non sia valida, ma soltanto che la chiave pubblica non è stata verificata.

Esecuzione del daemon

Esegui il daemon localmente dalla riga di comando. Utilizza l'opzione `-o` per eseguirlo in modalità locale e `-n` per impostare la regione.

```
~/Downloads$ ./xray -o -n us-east-2
```

Per ulteriori istruzioni specifiche per la piattaforma, consulta le seguenti sezioni:

- Linux (locale) — Esecuzione del demone X-Ray su Linux
- Windows (locale) — Esecuzione del demone X-Ray su Windows

- Elastic Beanstalk — Esecuzione del demone X-Ray su AWS Elastic Beanstalk
- Amazon EC2 — Esecuzione del demone X-Ray su Amazon EC2
- Amazon ECS — Esecuzione del daemon X-Ray su Amazon ECS

Puoi personalizzare ulteriormente il comportamento del daemon utilizzando le opzioni della riga di comando o un file di configurazione. Per informazioni dettagliate, vedi Configurazione del demone AWS X-Ray.

Dare al demone il permesso di inviare dati a X-Ray

Il demone X-Ray utilizza l' AWS SDK per caricare i dati di traccia su X-Ray e per farlo necessita AWS di credenziali con autorizzazione.

Su Amazon EC2, il daemon utilizza automaticamente il ruolo del profilo dell'istanza. Per informazioni sulle credenziali richieste per eseguire il demone localmente, consulta Esecuzione dell'applicazione localmente.

Se specifichi le credenziali in più di una posizione (file delle credenziali, profilo dell'istanza oppure variabili di ambiente), le credenziali utilizzate sono determinate dalla catena del provider dell'SDK. Per ulteriori informazioni sulla fornitura di credenziali all'SDK, consulta Specificing Credentials nella SDK for AWS Go Developer Guide.

Il ruolo o l'utente IAM a cui appartengono le credenziali del daemon devono avere l'autorizzazione di scrivere i dati sul servizio per tuo conto.

- Per utilizzare il daemon su Amazon EC2, crea un nuovo ruolo del profilo di istanza o aggiungi la policy gestita a una esistente.
- Per utilizzare il daemon su Elastic Beanstalk, aggiungi la policy gestita al ruolo del profilo di istanza predefinito di Elastic Beanstalk.
- Per eseguire il daemon localmente, consulta Esecuzione dell'applicazione localmente.

Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per AWS X-Ray.

Registri dei daemon X-Ray

Il demone emette informazioni sulla sua configurazione corrente e sui segmenti a cui invia. 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)
```

Per impostazione predefinita, il daemon invia i log in output su STDOUT. Se esegui il daemon in background, utilizza l'opzione della riga di comando `--log-file` o un file di configurazione per impostare il percorso del file di log. Puoi anche impostare il livello di tracciamento e disabilitare la rotazione dei log. Per istruzioni, consulta Configurazione del demone AWS X-Ray.

Su Elastic Beanstalk, la piattaforma imposta la posizione dei log del demone. Per informazioni dettagliate, vedi Esecuzione del demone X-Ray su AWS Elastic Beanstalk.

Configurazione del demone AWS X-Ray

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile utilizzare le opzioni della riga di comando o un file di configurazione per personalizzare il comportamento del demone X-Ray. La maggior parte delle opzioni sono disponibili utilizzando entrambi i metodi, ma alcune sono disponibili solo per i file di configurazione e alcune solo utilizzando la riga di comando.

Per iniziare, l'unica opzione che devi conoscere è `-n` or`--region`, che usi per impostare la regione che il demone usa per inviare i dati di traccia a X-Ray.

```
~/xray-daemon$ ./xray -n us-east-2
```

Se esegui il daemon localmente, cioè non su Amazon EC2, puoi aggiungere `-o` l'opzione per saltare il controllo delle credenziali del profilo, ad esempio, in modo che il daemon sia pronto più rapidamente.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Il resto delle opzioni della riga di comando consentono di configurare le opzioni di log, ascoltare su una porta diversa, limitare la quantità di memoria che il daemon può utilizzare o assumere un ruolo per l'invio dei dati di tracciamento a un altro account.

È possibile passare un file di configurazione al demone per accedere a opzioni di configurazione avanzate e fare cose come limitare il numero di chiamate simultanee a X-Ray, disabilitare la rotazione dei log e inviare traffico a un proxy.

Sections

- Variabili di ambiente supportate
- Utilizzo delle opzioni della riga di comando
- Utilizzo di un file di configurazione

Variabili di ambiente supportate

Il demone X-Ray supporta le seguenti variabili di ambiente:

- `AWS_REGION`— specifica l'endpoint Regione AWSdel servizio X-Ray.
- `HTTPS_PROXY`— Speciifica un indirizzo proxy per il demone tramite il quale caricare i segmenti. Può trattarsi dei nomi di dominio DNS o degli indirizzi IP e dei numeri di porta utilizzati dai server proxy.

Utilizzo delle opzioni della riga di comando

Passa queste opzioni al daemon quando lo esegui in locale o tramite un script dati utente.

Opzioni della riga di comando

- `-b`, `--bind` — Ascolta i documenti dei segmenti su una porta UDP diversa.

```
--bind "127.0.0.1:3000"
```

Predefinito:2000.

- `-t`, `--bind-tcp` — Ascolta le chiamate al servizio X-Ray su una porta TCP diversa.

```
-bind-tcp "127.0.0.1:3000"
```

Predefinito:. 2000

- `-c`, `--config` — Carica un file di configurazione dal percorso specificato.

```
--config "/home/ec2-user/xray-daemon.yaml"
```

- `-f`, `--log-file` — Registri di output nel percorso del file specificato.

```
--log-file "/var/log/xray-daemon.log"
```

- `-l`, `--log-level` — Livello di log, dal più dettagliato al meno dettagliato: dev, debug, info, warn, error, prod.

```
--log-level warn
```

Predefinito: `prod`

- `-m`, `--buffer-memory` — Modifica la quantità di memoria in MB che i buffer possono utilizzare (minimo 3).

```
--buffer-memory 50
```

Impostazione predefinita: 1% della memoria disponibile.

- `-o`, `--local-mode` — Non controllate i metadati delle EC2 istanze.
- `-r`, `--role-arn` — Assumi il ruolo IAM specificato per caricare i segmenti su un altro account.

```
--role-arn "arn:aws:iam::123456789012:role/xray-cross-account"
```

- `-a`, `--resource-arn` — Amazon Resource Name (ARN) della AWS risorsa che esegue il daemon.
- `-p`, `--proxy-address` — Carica i segmenti tramite un proxy. AWS X-Ray È necessario specificare il protocollo del server proxy.

```
--proxy-address "http://192.0.2.0:3000"
```

- `-n`, `--region` — Invia segmenti al servizio X-Ray in una regione specifica.
- `-v`, `--version` — Mostra la versione del AWS X-Ray demone.
- `-h`, `--help` — Mostra la schermata di aiuto.

Utilizzo di un file di configurazione

Per configurare il daemon puoi anche utilizzare un file in formato YAML. Trasferisci il file di configurazione al daemon utilizzando l'opzione `-c`.

```
~$ ./xray -c ~/xray-daemon.yaml
```

Opzioni file di configurazione

- `TotalBufferSizeMB`— Dimensione massima del buffer in MB (minimo 3). Scegli 0 per utilizzare l'1% della memoria dell'host.
- `Concurrency`— Numero massimo di chiamate simultanee per caricare documenti AWS X-Ray di segmento.
- `Region`— Invia segmenti al AWS X-Ray servizio in una regione specifica.
- `Socket`— Configura l'associazione del demone.
  - `UDPAddress`— Cambia la porta su cui il demone ascolta.
  - `TCPAddress`— Ascolta le chiamate al servizio X-Ray su una porta TCP diversa.
- `Logging`— Configura il comportamento di registrazione.
  - `LogRotation`— Impostato per `false` disabilitare la rotazione dei log.
  - `LogLevel`— Modifica il livello di registro, dal più dettagliato al meno:`dev`,`debug`, `info` o`prod`, `warnerror`,`prod`. L'impostazione predefinita è`prod`, che è equivalente a`info`.
  - `LogPath`— Registri di output nel percorso del file specificato.
- `LocalMode`— Impostato per `true` ignorare il controllo dei metadati, EC2 ad esempio.
- `ResourceARN`— Amazon Resource Name (ARN) della AWS risorsa che esegue il daemon.
- `RoleARN`— Assumi il ruolo IAM specificato per caricare i segmenti su un altro account.
- `ProxyAddress`— Carica i segmenti AWS X-Ray tramite un proxy.

- `Endpoint`— Modificare l'endpoint del servizio X-Ray a cui il demone invia i documenti del segmento.
- `NoVerifySSL`— Disabilita la verifica del certificato TLS.
- `Version`— Versione del formato del file di configurazione del demone. La versione del formato del file è un campo obbligatorio.

Example Xray-daemon.yaml

Questo file di configurazione modifica la porta di ascolto del daemon a 3000, disattiva i controlli dei metadati dell'istanza, imposta un ruolo da utilizzare per il caricamento dei segmenti e modifica regione e opzioni di logging.

```
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
```

Esecuzione locale del demone X-Ray

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi eseguire il AWS X-Ray demone localmente su Linux, macOS, Windows o in un contenitore Docker. Esegui il daemon per inoltrare i dati di traccia a X-Ray durante lo sviluppo e il test della tua

applicazione strumentata. Scarica ed estrai il daemon utilizzando le istruzioni contenute in questa pagina.

Quando viene eseguito localmente, il daemon può leggere le credenziali da un file di credenziali AWS SDK (`.aws/credentials`nella directory utente) o dalle variabili di ambiente. Per ulteriori informazioni, consulta Dare al demone il permesso di inviare dati a X-Ray.

Il daemon resta in attesa dei dati UDP sulla porta 2000. Puoi modificare la porta e altre opzioni utilizzando un file di configurazione e le opzioni della riga di comando. Per ulteriori informazioni, consulta Configurazione del demone AWS X-Ray.

Esecuzione del demone X-Ray su Linux

Puoi avviare il file eseguibile del daemon dalla riga di comando. Utilizza l'opzione `-o` per eseguirlo in modalità locale e `-n` per impostare la regione.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Per eseguire il daemon in background, utilizza &.

```
~/xray-daemon$ ./xray -o -n us-east-2 &
```

Termina un processo daemon in esecuzione in background `pkill`.

```
~$ pkill xray
```

Esecuzione del demone X-Ray in un contenitore Docker

Per eseguire il daemon in locale in un container Docker, salva il testo seguente in un file denominato `Dockerfile`. Scarica l'immagine di esempio completa su Amazon ECR. Per ulteriori informazioni, consulta la sezione Download del demone.

Example Docker file — 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
```

Crea l'immagine del container `docker build`.

```
~/xray-daemon$ docker build -t xray-daemon .
```

Esegui l'immagine in un container con `docker run`.

```
~/xray-daemon$ docker run \
      --attach STDOUT \
      -v ~/.aws/:/root/.aws/:ro \
      --net=host \
      -e AWS_REGION=us-east-2 \
      --name xray-daemon \
      -p 2000:2000/udp \
      xray-daemon -o
```

Questo comando utilizza le seguenti opzioni:

- `--attach STDOUT`— Visualizza l'output del demone nel terminale.
- `-v ~/.aws/:/root/.aws/:ro`— Concedi al contenitore l'accesso in sola lettura alla `.aws` directory per consentirgli di leggere le tue credenziali SDK. AWS
- `AWS_REGION=`*`us-east-2`*— Imposta la variabile di `AWS_REGION` ambiente per indicare al demone quale regione usare.
- `--net=host`— Collega il contenitore alla `host` rete. I containers sulla rete dell'host possono comunicare tra loro senza pubblicare le porte.
- `-p 2000:2000/udp`— Mappa la porta UDP 2000 del computer sulla stessa porta del contenitore. Questo non è necessaria per la comunicazione tra container nella stessa rete, ma consente di inviare segmenti al daemon dalla riga di comando o da un'applicazione non in esecuzione su Docker.
- `--name xray-daemon`— Assegna un nome al contenitore `xray-daemon` invece di generare un nome casuale.
- `-o`(dopo il nome dell'immagine): aggiunge l'`-o`opzione al punto di ingresso che esegue il demone all'interno del contenitore. Questa opzione indica al demone di funzionare in modalità locale per impedirgli di leggere i metadati delle EC2 istanze Amazon.

Per arrestare il daemon, utilizza `docker stop`. Se effettui modifiche al file `Dockerfile` e crei una nuova immagine, devi cancellare il container esistente prima di poterne creare un altro con lo stesso nome. Per eliminare il container, utilizza `docker rm`.

```
$ docker stop xray-daemon
$ docker rm xray-daemon
```

Esecuzione del demone X-Ray su Windows

Puoi avviare il file eseguibile del daemon dalla riga di comando. Utilizza l'opzione `-o` per eseguirlo in modalità locale e `-n` per impostare la regione.

```
> .\xray_windows.exe -o -n us-east-2
```

Usa uno PowerShell script per creare ed eseguire un servizio per il demone.

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
```

Esecuzione del demone X-Ray su OS X

Puoi avviare il file eseguibile del daemon dalla riga di comando. Utilizza l'opzione `-o` per eseguirlo in modalità locale e `-n` per impostare la regione.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2
```

Per eseguire il daemon in background, utilizza &.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2 &
```

Utilizza nohup per evitare che il daemon venga terminato alla chiusura del terminale.

```
~/xray-daemon$ nohup ./xray_mac &
```

Esecuzione del demone X-Ray su AWS Elastic Beanstalk

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Per inoltrare i dati di traccia dall'applicazione a AWS X-Ray, puoi eseguire il daemon X-Ray sulle istanze Amazon EC2 del tuo ambiente Elastic Beanstalk. Per un elenco delle piattaforme supportate, consulta Configuring Debugging nella Developer Guide. AWS X-Ray AWS Elastic Beanstalk

> **Note**
>
> Il daemon utilizza il profilo dell'istanza dell'ambiente per acquisire le autorizzazioni. Per istruzioni sull'aggiunta di autorizzazioni al profilo dell'istanza Elastic Beanstalk, consulta. Dare al demone il permesso di inviare dati a X-Ray

Le piattaforme Elastic Beanstalk forniscono un'opzione di configurazione che puoi impostare per eseguire il demone automaticamente. Puoi abilitare il demone in un file di configurazione nel codice sorgente o scegliendo un'opzione nella console Elastic Beanstalk. Quando abiliti l'opzione di configurazione, il daemon viene installato sull'istanza e viene eseguito come servizio.

La versione inclusa nelle piattaforme Elastic Beanstalk potrebbe non essere la versione più recente. Consulta la sezione Piattaforme Supportate per scoprire la versione del daemon che è disponibile per la configurazione della tua piattaforma.

Elastic Beanstalk non fornisce il daemon X-Ray sulla piattaforma Multicontainer Docker (Amazon ECS).

Utilizzo dell'integrazione X-Ray di Elastic Beanstalk per eseguire il demone X-Ray

Usa la console per attivare l'integrazione X-Ray o configurala nel codice sorgente dell'applicazione con un file di configurazione.

Per abilitare il demone X-Ray nella console Elastic Beanstalk

1. Apri la console Elastic Beanstalk.
2. Accedi alla console di gestione del tuo ambiente.
3. Scegliere Configuration (Configurazione).
4. Scegliere Software Settings (Impostazioni software)
5. Alla voce X-Ray Daemon (Daemon X-Ray), scegliere Enabled (Abilitato).
6. Scegli Applica.

Per rendere la configurazione portabile tra più ambienti puoi includere un file di configurazione nel codice sorgente.

Example.ebextensions/xray-daemon.config

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

Elastic Beanstalk passa un file di configurazione al demone e invia i log in una posizione standard.

Sulle piattaforme Windows Server

- File di configurazione — `C:\Program Files\Amazon\XRay\cfg.yaml`
- Registri — `c:\Program Files\Amazon\XRay\logs\xray-service.log`

Sulle piattaforme Linux

- File di configurazione — `/etc/amazon/xray/cfg.yaml`
- Registri — `/var/log/xray/xray.log`

Elastic Beanstalk fornisce strumenti per estrarre i log delle istanze dalla riga di comando o. Console di gestione AWS Puoi dire a Elastic Beanstalk di includere i log dei daemon X-Ray aggiungendo un'attività con un file di configurazione.

Example.ebextensions/xray-logs.config - Linux

```
files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      /var/log/xray/xray.log
```

Example.ebextensions/xray-logs.config - Windows Server

```
files:
  "c:/Program Files/Amazon/ElasticBeanstalk/config/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      c:\Progam Files\Amazon\XRay\logs\xray-service.log
```

Per ulteriori informazioni, consulta Visualizzazione dei log dalle istanze Amazon EC2 del tuo ambiente Elastic Beanstalk nella Developer AWS Elastic Beanstalk Guide.

Scaricamento ed esecuzione manuale del demone X-Ray (avanzato)

Se il demone X-Ray non è disponibile per la configurazione della tua piattaforma, puoi scaricarlo da Amazon S3 ed eseguirlo con un file di configurazione.

Usa un file di configurazione Elastic Beanstalk per scaricare ed eseguire il demone.

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
```

Questi esempi aggiungono anche il file di registro del demone all'attività Elastic Beanstalk tail logs, in modo che sia incluso quando richiedi i log con la console o l'Elastic Beanstalk Command Line Interface (EB CLI).

Esecuzione del demone X-Ray su Amazon EC2

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi eseguire il daemon X-Ray sui seguenti sistemi operativi su Amazon: EC2

- Amazon Linux
- Ubuntu
- Windows Server (2012 R2 e versioni successive)

Utilizzate un profilo di istanza per concedere al demone l'autorizzazione a caricare i dati di traccia su X-Ray. Per ulteriori informazioni, consulta Dare al demone il permesso di inviare dati a X-Ray.

Utilizza uno script dati utente per eseguire automaticamente il daemon all'avvio dell'istanza.

Example Script di dati utente - 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 dati utente - 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>
```

Esecuzione del daemon X-Ray su Amazon ECS

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

In Amazon ECS, crea un'immagine Docker che esegua il daemon X-Ray, caricala in un repository di immagini Docker e quindi distribuiscila nel tuo cluster Amazon ECS. Puoi utilizzare la mappatura delle

porte e la modalità di impostazione della rete nel file di definizione dell'attività per consentire alla tua applicazione di comunicare con il container del daemon.

Utilizzo dell'immagine Docker ufficiale

X-Ray fornisce un'immagine del contenitore Docker su Amazon ECR che puoi distribuire insieme alla tua applicazione. Per ulteriori informazioni, consulta la sezione Download del daemon.

Example Definizione di attività

```
{
  "name": "xray-daemon",
  "image": "amazon/aws-xray-daemon",
  "cpu": 32,
  "memoryReservation": 256,
  "portMappings" : [
      {
          "hostPort": 0,
          "containerPort": 2000,
          "protocol": "udp"
      }
   ]
}
```

Creare e compilare un'immagine Docker

Per la configurazione personalizzata, potrebbe essere necessario definire la propria immagine Docker.

Aggiungi policy gestite al tuo ruolo di attività per concedere al demone l'autorizzazione a caricare i dati di traccia su X-Ray. Per ulteriori informazioni, consulta Dare al demone il permesso di inviare dati a X-Ray.

Utilizza uno dei seguenti Dockerfiles per creare un'immagine che esegue il daemon.

Example Docker file — 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**
>
> I contrassegni `-t` e `-b` sono necessari per specificare un indirizzo di binding per ascoltare il loopback di un ambiente multi-container.

Example Dockerfile — Ubuntu

Per le derivazioni di Debian, è inoltre necessario installare i certificati della Certificate Autohority (CA) per evitare problemi durante il download del programma di installazione.

```
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
```

Nella tua definizione di attività, la configurazione dipende dalla modalità di rete utilizzata. La modalità di rete bridge è l'impostazione predefinita e può essere utilizzato nel VPC di default. In una rete bridge, imposta la variabile di `AWS_XRAY_DAEMON_ADDRESS` ambiente per indicare all'SDK X-Ray a quale porta contenitore fare riferimento e imposta la porta host. In una rete bridge, pubblica la porta UDP 2000 e crea un collegamento dal container della tua applicazione al container del daemon.

Example Definizione di attività

```
    {
      "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 esegui il cluster nella sottorete privata di un VPC, puoi utilizzare la modalità di rete `awsvpc` per collegare ai tuoi container un'interfaccia di rete elastica (ENI). In questo modo puoi evitare di utilizzare i collegamenti. Ometti la porta dell'host nel mappaggio delle porte, il collegamento e la variabile di ambiente `AWS_XRAY_DAEMON_ADDRESS`.

Example Definizione dell'attività del 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
                }
            ]
        }
    ]
}
```

Configura le opzioni della riga di comando nella console Amazon ECS

Le opzioni della riga di comando sovrascrivono qualsiasi valore in conflitto nel file di configurazione dell'immagine. Le opzioni della riga di comando sono in genere utilizzate per i test locali, ma possono anche essere usate per comodità durante l'impostazione delle variabili di ambiente o per controllare il processo di avvio.

Aggiungendo le opzioni della riga di comando, aggiorni il Docker `CMD` che viene passato al contenitore. Per ulteriori informazioni, consulta la pagina di riferimento per l'esecuzione del Docker.

Come impostare un'opzione della riga di comando

1. Apri la console Amazon ECS classic all'indirizzo https://console.aws.amazon.com/ecs/.

2. Dalla barra di navigazione, scegli la regione in cui si trova la definizione di attività.
3. Nel riquadro di navigazione, scegli Definizioni di attività.
4. Nella pagina Task Definitions (Definizioni di attività), seleziona la casella a sinistra della definizione di attività da modificare e scegli Create new revision (Crea nuova revisione).
5. Nella pagina Create new revision of Task Definition (Crea nuova revisione della definizione di attività) selezionare il contenitore.
6. Nella sezione ENVIRONMENT (AMBIENTE) aggiungere l'elenco separato da virgole di opzioni della riga di comando al campo Command (Comando).
7. Scegliere Aggiorna.
8. Verifica le informazioni e scegli Crea.

Nell'esempio seguente viene illustrato come scrivere un'opzione della riga di comando separata da virgole per l'opzione `RoleARN`. L'`RoleARN`opzione presuppone il ruolo IAM specificato per caricare i segmenti su un account diverso.

Example

```
--role-arn, arn:aws:iam::123456789012:role/xray-cross-account
```

Per ulteriori informazioni sulle opzioni della riga di comando disponibili in X-Ray, vedere Configurazione del demone. AWS X-Ray

Integrazione AWS X-Ray con altri Servizi AWS

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Molti Servizi AWS offrono diversi livelli di integrazione con i raggi X, tra cui il campionamento e l'aggiunta di intestazioni alle richieste in arrivo, l'esecuzione del demone X-Ray e l'invio automatico di dati di traccia a X-Ray. L'integrazione con X-Ray può includere quanto segue:

- Strumentazione attiva: richieste in arrivo di campioni e strumenti
- Strumentazione passiva: richieste di strumenti che sono state campionate da un altro servizio
- Tracciamento delle richieste: aggiunge un'intestazione di tracciamento a tutte le richieste in arrivo e la propaga a valle
- Strumenti: esegue il demone X-Ray per ricevere segmenti dall'SDK X-Ray

> **Note**
>
> X-Ray SDKs include plugin per un'ulteriore integrazione con. Servizi AWS Ad esempio, puoi utilizzare il plug-in X-Ray SDK for Java Elastic Beanstalk per aggiungere informazioni sull'ambiente Elastic Beanstalk che esegue l'applicazione, inclusi il nome e l'ID dell'ambiente.

Ecco alcuni esempi Servizi AWS integrati con X-Ray:

- AWS Distro for OpenTelemetry (ADOT): con ADOT, gli ingegneri possono strumentare le proprie applicazioni una sola volta e inviare metriche e tracce correlate a più AWS soluzioni di monitoraggio tra cui Amazon CloudWatch, AWS X-Ray Amazon Service e Amazon Managed Service for OpenSearch Prometheus.

- AWS Lambda— Strumentazione attiva e passiva delle richieste in arrivo in tutte le fasi di esecuzione. AWS Lambda aggiunge due nodi alla mappa di traccia, uno per il AWS Lambda servizio e uno per la funzione. Quando si abilita la strumentazione, esegue AWS Lambda anche il demone X-Ray su Java e i runtime Node.js da utilizzare con l'SDK X-Ray.
- Amazon API Gateway: strumentazione attiva e passiva. API Gateway utilizza regole di campionamento per determinare quali richieste registrare e aggiunge un nodo per la fase gateway alla mappa dei servizi.
- AWS Elastic Beanstalk— Utensili. Elastic Beanstalk include il demone X-Ray sulle seguenti piattaforme:
  - Java SE — 2.3.0 e configurazioni successive
  - Tomcat — 2.4.0 e configurazioni successive
  - Node.js — 3.2.0 e configurazioni successive
  - Windows Server: tutte le configurazioni diverse da Windows Server Core rilasciate dopo il 9 dicembre 2016

  Puoi usare la console Elastic Beanstalk per dire a Elastic Beanstalk di eseguire il demone su queste piattaforme oppure utilizzare l'opzione nel namespace. `XRayEnabled` `aws:elasticbeanstalk:xray`
- Elastic Load Balancing: traccia delle richieste sugli Application Load Balancer. L'Application Load Balancer aggiunge l'ID di traccia all'intestazione della richiesta prima di inviarlo a un gruppo target.
- Amazon EventBridge — Strumentazione passiva. Se un servizio che pubblica eventi su EventBridge è dotato di strumentazione con X-Ray SDK, le destinazioni degli eventi riceveranno l'intestazione di tracciamento e potranno continuare a propagare l'ID di traccia originale.
- Amazon Simple Notification Service — Strumentazione passiva. Se un publisher Amazon SNS traccia il proprio client con l'SDK X-Ray, gli abbonati possono recuperare l'intestazione di tracciamento e continuare a propagare la traccia originale dell'editore con lo stesso ID di traccia.
- Amazon Simple Queue Service: strumentazione passiva. Se un servizio traccia le richieste utilizzando l'SDK X-Ray, Amazon SQS può inviare l'intestazione di tracciamento e continuare a propagare la traccia originale dal mittente al consumatore con un ID di traccia coerente.
- Amazon Bedrock AgentCore: AgentCore supporta il tracciamento distribuito tramite l'integrazione a raggi X, consentendoti di tenere traccia delle richieste mentre fluiscono attraverso le tue applicazioni agente. Quando abiliti l'osservabilità AgentCore delle tue risorse, puoi diffondere il contesto di tracciamento oltre i confini del servizio e ottenere visibilità sulle prestazioni dei tuoi agenti e strumenti di intelligenza artificiale.

Scegli tra i seguenti argomenti per esplorare il set completo di soluzioni integrate. Servizi AWS

Argomenti

- 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 per OpenTelemetry e AWS X-Ray
- Monitoraggio delle modifiche alla configurazione della X-Ray crittografia con AWS Config
- AWS AppSync e AWS X-Ray
- Supporto di tracciamento attivo di Amazon API Gateway per AWS X-Ray
- Amazon EC2 e AWS App Mesh
- AWS App Runner e X-Ray
- Registrazione delle chiamate all'API X-Ray con AWS CloudTrail
- CloudWatch integrazione con X-Ray
- AWS Elastic Beanstalk e AWS X-Ray
- Elastic Load Balancing e AWS X-Ray
- Amazon EventBridge e AWS X-Ray
- AWS Lambda e AWS X-Ray
- AWS Step Functions and AWS X-Ray

Amazon Bedrock AgentCore e AWS X-Ray

Amazon Bedrock AgentCore si integra con AWS X-Ray per fornire funzionalità di tracciamento distribuite per i tuoi agenti e strumenti di intelligenza artificiale. Questa integrazione ti consente di tenere traccia delle richieste mentre fluiscono attraverso le applicazioni degli agenti, aiutandoti a identificare i colli di bottiglia nelle prestazioni e a risolvere i problemi.

AgentCore supporta il tracciamento distribuito tramite X-Ray integrazione, consentendoti di monitorare le prestazioni degli agenti e degli strumenti di intelligenza artificiale. Quando abiliti

l'osservabilità AgentCore delle tue risorse, puoi propagare il contesto di tracciamento oltre i confini dei servizi e ottenere visibilità sul modo in cui i tuoi agenti interagiscono con altri servizi. AWS Per maggiori informazioni, consulta Amazon Bedrock AgentCore.

AgentCore supporta le seguenti funzionalità: X-Ray

- Propagazione del contesto di traccia ai servizi a valle
- Strumentazione personalizzata utilizzando l'SDK AWS Distro for (ADOT) OpenTelemetry

X-Ray Configurazione con AgentCore

Per utilizzare X-Ray con AgentCore, devi abilitare CloudWatch Transaction Search nel tuo AWS account. Si tratta di una configurazione una tantum che consente di AgentCore inviare dati di tracciamento a X-Ray. Per ulteriori informazioni, consulta Abilitare la ricerca delle transazioni.

Per ulteriori informazioni sulla configurazione dell'osservabilità per AgentCore, consulta Aggiungere l'osservabilità al tuo AgentCore agente o strumento Amazon Bedrock.

Usare gli header di traccia con AgentCore

AgentCore supporta il formato di intestazione di X-Ray traccia per la traccia distribuita. È possibile includere l'`X-Amzn-Trace-Id`intestazione nelle richieste per mantenere il contesto di traccia oltre AgentCore i confini del servizio.

Amazon Elastic Compute Cloud e AWS X-Ray

Puoi installare ed eseguire il demone X-Ray su un' EC2 istanza Amazon con uno script di dati utente. Per istruzioni, consulta Esecuzione del demone X-Ray su Amazon EC2.

Utilizzate un profilo di istanza per concedere al demone l'autorizzazione a caricare i dati di traccia su X-Ray. Per ulteriori informazioni, consulta Dare al demone il permesso di inviare dati a X-Ray.

Amazon SNS e AWS X-Ray

Puoi utilizzarlo AWS X-Ray con Amazon Simple Notification Service (Amazon SNS) per tracciare e analizzare le richieste man mano che attraversano gli argomenti SNS e arrivano ai tuoi SNS-supported servizi in abbonamento. Usa il X-Ray tracciamento con Amazon SNS per analizzare

le latenze nei tuoi messaggi e nei relativi servizi di back-end, ad esempio quanto tempo trascorre una richiesta in un argomento e quanto tempo è necessario per recapitare il messaggio a ciascuno degli abbonamenti dell'argomento. Amazon SNS supporta X-Ray il tracciamento sia per argomenti standard che per argomenti FIFO.

Se pubblichi un argomento su Amazon SNS da un servizio già dotato di strumentazione X-Ray, Amazon SNS trasmette il contesto di tracciamento dall'editore agli abbonati. Inoltre, puoi attivare il tracciamento attivo per inviare dati di segmento relativi ai tuoi abbonamenti Amazon SNS X-Ray per i messaggi pubblicati da un client SNS strumentato. Attiva il tracciamento attivo per un argomento di Amazon SNS utilizzando la console Amazon SNS o utilizzando l'API o l'interfaccia a riga di comando di Amazon SNS. Per ulteriori informazioni sulla strumentazione dei client SNS, consulta Instrumenting your application.

Configura il tracciamento attivo di Amazon SNS

Puoi utilizzare la console Amazon SNS o l'interfaccia a riga di comando o l'SDK per configurare il AWS tracciamento attivo di Amazon SNS.

Quando utilizzi la console Amazon SNS, Amazon SNS tenta di creare le autorizzazioni necessarie per le chiamate da parte di SNS. X-Ray Il tentativo può essere rifiutato se non disponi di autorizzazioni sufficienti per modificare le politiche relative alle risorse. X-Ray Per ulteriori informazioni su queste autorizzazioni, consulta Gestione delle identità e degli accessi in Amazon SNS e Casi di esempio per il controllo degli accessi di Amazon SNS nella Amazon Simple Notification Service Developer Guide. Per ulteriori informazioni sull'attivazione dell'active tracing tramite la console Amazon SNS, consulta l'argomento Enabling active tracing on an Amazon SNS nella Amazon Simple Notification Service Developer Guide.

Quando utilizzi la AWS CLI o l'SDK per attivare l'active tracing, devi configurare manualmente le autorizzazioni utilizzando policy basate sulle risorse. Utilizzalo `PutResourcePolicy` per configurare X-Ray con la policy basata sulle risorse necessaria a cui consentire ad Amazon SNS di inviare tracce a. X-Ray

Example Esempio di policy basata sulle X-Ray risorse per l'active tracing di Amazon SNS

Questo documento di policy di esempio specifica le autorizzazioni necessarie ad Amazon SNS per inviare i dati di tracciamento a: X-Ray

```
{
    Version: "2012-10-17",
```

```
    Statement: [
      {
        Sid: "SNSAccess",
        Effect: Allow,
        Principal: {
          Service: "sns.amazonaws.com",
        },
        Action: [
          "xray:PutTraceSegments",
          "xray:GetSamplingRules",
          "xray:GetSamplingTargets"
        ],
        Resource: "*",
        Condition: {
          StringEquals: {
            "aws:SourceAccount": "account-id"
          },
          StringLike: {
            "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
          }
        }
      }
    ]
  }
```

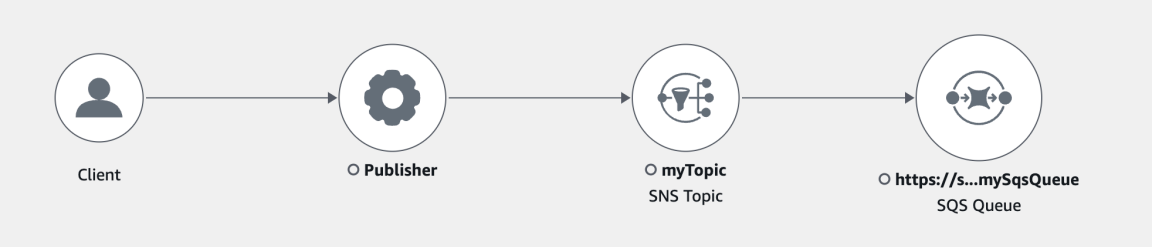
Utilizza l'interfaccia a riga di comando per creare una policy basata sulle risorse che conceda ad Amazon SNS le autorizzazioni per inviare dati di tracciamento a: X-Ray

```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
 '{ "Version": "2012-10-17",        "Statement": [ { "Sid": "SNSAccess",
 "Effect": "Allow", "Principal": { "Service": "sns.amazonaws.com" }, "Action":
 [ "xray:PutTraceSegments", "xray:GetSamplingRules", "xray:GetSamplingTargets" ],
 "Resource": "*", "Condition": { "StringEquals": { "aws:SourceAccount": "account-
id" }, "StringLike": { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-
name" } } } ] }'
```

Per utilizzare questi esempi, sostituisci, *partitionregion*, e *topic-name* con la tua AWS partizione*account-id*, regione, ID account e nome dell'argomento Amazon SNS specifici. Per autorizzare tutti gli argomenti di Amazon SNS a cui inviare dati di tracciamento X-Ray, sostituisci il nome dell'argomento con. *

Visualizza le tracce degli editori e degli abbonati di Amazon SNS nella console X-Ray

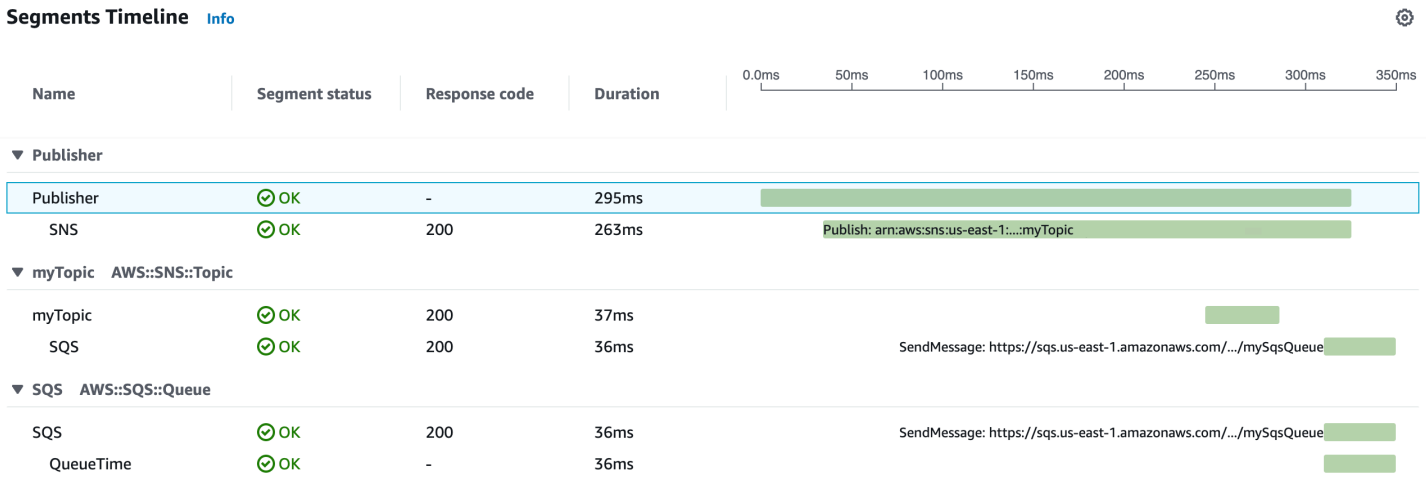
Usa la X-Ray console per visualizzare una mappa di tracciamento e i dettagli di tracciamento che mostrano una vista connessa degli editori e degli abbonati di Amazon SNS. Quando l'active tracing di Amazon SNS è attivato per un argomento, la mappa di traccia e la mappa dei dettagli di X-Ray traccia mostrano i nodi connessi per gli editori di Amazon SNS, l'argomento Amazon SNS e gli abbonati a valle:



Dopo aver scelto una traccia che includa un editore e un abbonato di Amazon SNS, la pagina dei dettagli della X-Ray traccia mostra una mappa dei dettagli della traccia e una cronologia del segmento.

Example Esempio di cronologia con editore e abbonato di Amazon SNS

Questo esempio mostra una sequenza temporale che include un publisher di Amazon SNS che invia un messaggio a un argomento Amazon SNS, che viene elaborato da un abbonato Amazon SQS.



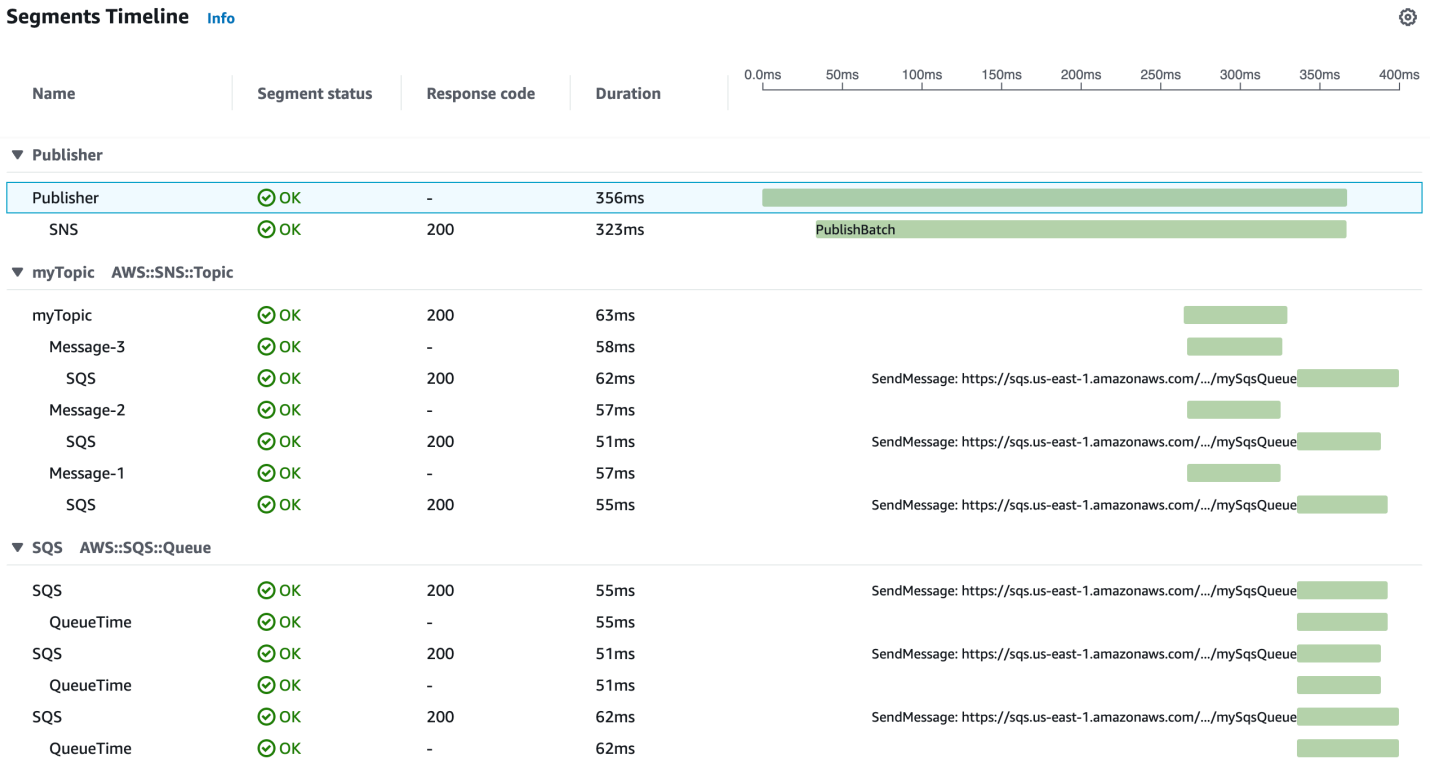
La sequenza temporale di esempio riportata sopra fornisce dettagli sul flusso di messaggi di Amazon SNS:

- Il segmento SNS rappresenta la durata di andata e ritorno della chiamata `Publish` API dal client.

- Il segmento MyTopic rappresenta la latenza della risposta di Amazon SNS alla richiesta di pubblicazione.
- Il sottosegmento SQS rappresenta il tempo di andata e ritorno impiegato da Amazon SNS per pubblicare il messaggio in una coda Amazon SQS.
- Il tempo che intercorre tra il segmento MyTopic e il sottosegmento SQS rappresenta il tempo di permanenza del messaggio nel sistema Amazon SNS.

Example Esempio di cronologia con messaggi Amazon SNS in batch

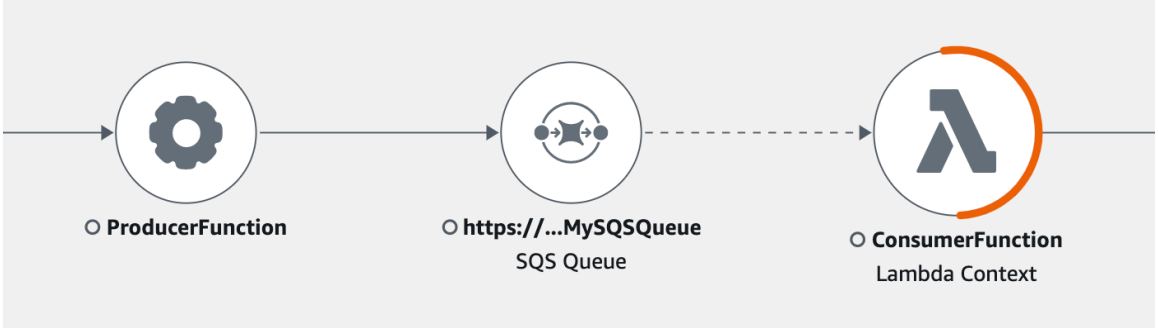
Se più messaggi Amazon SNS vengono raggruppati in batch all'interno di una singola traccia, la timeline del segmento mostra i segmenti che rappresentano ogni messaggio elaborato.



Amazon SQS e AWS X-Ray

AWS X-Ray si integra con Amazon Simple Queue Service (Amazon SQS) per tracciare i messaggi passati attraverso una coda Amazon SQS. Se un servizio traccia le richieste utilizzando l' X-Ray SDK, Amazon SQS può inviare l'intestazione di tracciamento e continuare a propagare la traccia originale dal mittente al consumatore con un ID di traccia coerente. Questa continuità consente agli utenti di tracciare, analizzare ed eseguire il debug di tutti i servizi di downstream.

AWS X-Ray supporta il tracciamento di applicazioni basate su eventi utilizzando Amazon SQS e. AWS Lambda Usa la CloudWatch console per visualizzare una vista connessa di ogni richiesta mentre viene accodata con Amazon SQS ed elaborata da una funzione Lambda downstream. Le tracce provenienti dai produttori di messaggi a monte vengono automaticamente collegate alle tracce provenienti dai nodi consumer Lambda a valle, creando una vista end-to-end dell'applicazione. Per ulteriori informazioni, consulta Tracing event-driven applications. ???



Amazon SQS supporta la seguente strumentazione per l'header di tracciamento:

- Intestazione HTTP predefinita: l' X-Ray SDK compila automaticamente l'intestazione di traccia come intestazione HTTP quando si chiama Amazon SQS tramite l'SDK. AWS L'intestazione di traccia predefinita viene trasportata da `X-Amzn-Trace-Id` e corrisponde a tutti i messaggi inclusi in una richiesta `SendMessage` o `SendMessageBatch`. Per ulteriori informazioni sull'intestazione HTTP predefinita, consulta Intestazione di tracciamento.
- **`AWSTraceHeader`**Attributo di sistema: `AWSTraceHeader` è un attributo di sistema di messaggi riservato da Amazon SQS per contenere l'intestazione di traccia con i X-Ray messaggi in coda. `AWSTraceHeader`è disponibile per l'uso anche quando la strumentazione automatica tramite l'SDK non lo è, ad esempio quando si crea un X-Ray SDK di tracciamento per una nuova lingua. Quando sono impostate entrambe le strumentazioni delle intestazioni, l'attributo del sistema di messaggi sostituisce l'intestazione di traccia HTTP.

Quando viene eseguito su Amazon EC2, Amazon SQS supporta l'elaborazione di un messaggio alla volta. Ciò si applica quando viene eseguito su un host locale e quando si utilizzano servizi container AWS Fargate, come Amazon ECS o. AWS App Mesh

L'intestazione di traccia è esclusa sia dalla dimensione dei messaggi di Amazon SQS che dalle quote degli attributi dei messaggi. L'attivazione del X-Ray tracciamento non supererà le tue quote Amazon SQS. Per ulteriori informazioni sulle AWS quote, consulta Amazon SQS Quotas.

Invio dell'intestazione di traccia HTTP

I componenti del mittente in Amazon SQS possono inviare automaticamente l'intestazione di traccia tramite la chiamata or. `SendMessageBatch` `SendMessage` Quando i client AWS SDK sono strumentati, possono essere tracciati automaticamente in tutte le lingue supportate dall'SDK. X-Ray I tracciati Servizi AWS e le risorse a cui accedi all'interno di tali servizi (ad esempio, un bucket Amazon S3 o una coda Amazon SQS) vengono visualizzati come nodi a valle sulla mappa di traccia nella console. X-Ray

Per sapere come tracciare le chiamate AWS SDK con la tua lingua preferita, consulta i seguenti argomenti negli SDK supportati:

- Go: Tracciamento delle chiamate AWS SDK con X-Ray SDK for Go
- Java: Tracciamento delle chiamate AWS SDK con X-Ray SDK for Java
- Node.js – Tracciamento delle chiamate AWS SDK con X-Ray SDK per Node.js
- Python: Tracciamento delle chiamate AWS SDK con X-Ray SDK per Python
- Ruby: Tracciamento delle chiamate AWS SDK con X-Ray SDK for Ruby
- .NET: Tracciamento delle chiamate AWS SDK con X-Ray SDK for .NET

Recupero dell'intestazione di traccia e recupero del contesto di traccia

Se utilizzi un consumer Lambda downstream, la propagazione del contesto di traccia è automatica. Per continuare la propagazione del contesto con altri consumatori Amazon SQS, devi strumentare manualmente l'handoff al componente ricevente.

Esistono tre fasi principali per recuperare il contesto di traccia:

- Ricezione del messaggio dalla coda per l'attributo `AWSTraceHeader` chiamando l'API `ReceiveMessage`.
- Recupero dell'intestazione di traccia dall'attributo.
- Recupero dell'ID di traccia dall'intestazione. Facoltativamente, aggiunta di ulteriori parametri al segmento.

Di seguito è riportato un esempio di implementazione scritta con l' X-Ray SDK per Java.

Example: recupero dell'intestazione di traccia e recupero del contesto di traccia

```
// 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

AWS X-Ray si integra con Amazon S3 per tracciare le richieste upstream per aggiornare i bucket S3 dell'applicazione. Se un servizio traccia le richieste utilizzando l'SDK X-Ray, Amazon S3 può inviare le intestazioni di tracciamento a sottoscrittori di eventi downstream come Amazon SQS e Amazon SNS. AWS Lambda X-Ray abilita i messaggi di tracciamento per le notifiche di eventi di Amazon S3.

È possibile utilizzare la mappa di traccia X-Ray per visualizzare le connessioni tra Amazon S3 e altri servizi utilizzati dall'applicazione. È inoltre possibile utilizzare la console per visualizzare i parametri come la latenza media e le percentuali di errore. Per ulteriori informazioni sulla console X-Ray, vedere. Usa la console X-Ray

Amazon S3 supporta la strumentazione di intestazione http predefinita. L'SDK X-Ray compila automaticamente l'intestazione di traccia come intestazione HTTP quando chiami Amazon S3 tramite l'SDK. AWS L'intestazione di traccia predefinita è gestita da. `X-Amzn-Trace-Id` Per ulteriori

informazioni sul tracciamento delle intestazioni, consulta la Intestazione di tracciamento pagina concettuale. La propagazione del contesto di traccia di Amazon S3 supporta i seguenti abbonati: Lambda, SQS e SNS. Poiché SQS e SNS non emettono autonomamente i dati dei segmenti, non verranno visualizzati nella traccia o nella mappa di traccia quando vengono attivati da S3, anche se propagheranno l'intestazione di tracciamento ai servizi a valle.

Configurazione delle notifiche degli eventi di Amazon S3

Con la funzione di notifica di Amazon S3, ricevi notifiche quando si verificano determinati eventi nel tuo bucket. Queste notifiche possono quindi essere propagate alle seguenti destinazioni all'interno dell'applicazione:

- Servizio di notifica semplice Amazon (Amazon Simple Notification Service (Amazon SNS))
- Amazon Simple Queue Service (Amazon SQS)
- AWS Lambda

Per un elenco degli eventi supportati, consulta Tipi di eventi supportati nella guida per sviluppatori di Amazon S3.

Amazon SNS e Amazon SQS

Per pubblicare notifiche su un argomento SNS o una coda SQS, devi prima concedere le autorizzazioni ad Amazon S3. Per concedere queste autorizzazioni, alleghi una policy AWS Identity and Access Management (IAM) all'argomento SNS o alla coda SQS di destinazione. Per ulteriori informazioni sulle policy IAM richieste, consulta Concessione delle autorizzazioni per pubblicare messaggi su un argomento SNS o su una coda SQS.

Per informazioni sull'integrazione di SNS e SQS con X-Ray, vedere, e. Amazon SNS e AWS X-Ray Amazon SQS e AWS X-Ray

AWS Lambda

Quando usi la console Amazon S3 per configurare le notifiche di eventi su un bucket S3 per una funzione Lambda, la console imposta le autorizzazioni necessarie sulla funzione Lambda in modo che Amazon S3 disponga delle autorizzazioni per richiamare la funzione dal bucket. Per ulteriori informazioni, consulta Come posso abilitare e configurare le notifiche di eventi per un bucket S3? nella Guida per l'utente della console di Amazon Simple Storage Service.

Puoi anche concedere ad Amazon S3 le autorizzazioni AWS Lambda per richiamare la tua funzione Lambda. Per ulteriori informazioni, consulta Tutorial: Using AWS Lambda with Amazon S3 nella AWS Lambda Developer Guide.

Per ulteriori informazioni sull'integrazione di Lambda con X-Ray, consulta Strumentazione del codice Java in Lambda. AWS

AWS Distro per OpenTelemetry e AWS X-Ray

Usa AWS Distro for OpenTelemetry (ADOT) per raccogliere e inviare metriche e tracce verso AWS X-Ray e altre soluzioni di monitoraggio, come Amazon, Amazon OpenSearch Service e CloudWatch Amazon Managed Service for Prometheus.

AWS Distro per OpenTelemetry

The AWS Distro for OpenTelemetry (ADOT) è una AWS distribuzione basata sul progetto Cloud Native Computing Foundation (CNCF). OpenTelemetry OpenTelemetry fornisce un unico set di librerie APIs, agenti e librerie open source per raccogliere tracce e metriche distribuite. Questo toolkit è una distribuzione di OpenTelemetry componenti upstream SDKs, inclusi agenti di strumentazione automatica e collettori testati, ottimizzati, protetti e supportati da. AWS

Con ADOT, gli ingegneri possono strumentare le loro applicazioni una sola volta e inviare metriche e tracce correlate a più soluzioni di AWS monitoraggio tra cui Amazon CloudWatch, Amazon OpenSearch Service e AWS X-Ray Amazon Managed Service for Prometheus.

ADOT è integrato con un numero crescente di strumenti Servizi AWS per semplificare l'invio di tracce e metriche a soluzioni di monitoraggio come X-Ray. Alcuni esempi di servizi integrati con ADOT includono:

- AWS Lambda— i layer Lambda AWS gestiti per ADOT offrono un'esperienza plug-and-play utente strumentando automaticamente una funzione Lambda, OpenTelemetry unitamente a una out-of-the-box configurazione per AWS Lambda X-Ray in un layer facile da configurare. Gli utenti possono abilitare e disabilitare OpenTelemetry la funzione Lambda senza modificare il codice. Per ulteriori informazioni, consulta AWS Distro for OpenTelemetry Lambda
- Amazon Elastic Container Service (ECS): raccogli metriche e tracce dalle applicazioni Amazon ECS utilizzando AWS Distro for OpenTelemetry Collector, per inviarle a X-Ray e altre soluzioni di monitoraggio. Per ulteriori informazioni, consulta la sezione Raccolta dei dati di tracciamento delle applicazioni nella guida per sviluppatori di Amazon ECS.

- AWS App Runner — App Runner supporta l'invio di tracce a X-Ray utilizzando AWS Distro for (ADOT). OpenTelemetry Usa ADOT per SDKs raccogliere dati di traccia per le tue applicazioni containerizzate e usa X-Ray per analizzare e ottenere informazioni dettagliate sulla tua applicazione strumentata. Per ulteriori informazioni, vedere AWS App Runner e X-Ray.

Per ulteriori informazioni su AWS Distro for OpenTelemetry, inclusa l'integrazione con altre Servizi AWS, consulta AWS Distro for Documentation. OpenTelemetry

Per ulteriori informazioni sulla strumentazione dell'applicazione con AWS Distro for e OpenTelemetry X-Ray, vedere Strumentazione dell'applicazione con Distro for. AWS OpenTelemetry

Monitoraggio delle modifiche alla configurazione della X-Ray crittografia con AWS Config

AWS X-Ray si integra AWS Config per registrare le modifiche alla configurazione apportate alle risorse di X-Ray crittografia. È possibile utilizzarlo AWS Config per inventariare le risorse di X-Ray crittografia, controllare la cronologia delle X-Ray configurazioni e inviare notifiche in base alle modifiche delle risorse.

AWS Config supporta la registrazione delle seguenti modifiche alle risorse di X-Ray crittografia come eventi:

- Modifiche alla configurazione: modifica o aggiunta di una chiave di crittografia o ripristino dell'impostazione di X-Ray crittografia predefinita.

Usa le seguenti istruzioni per imparare a creare una connessione di base tra X-Ray e AWS Config.

Creazione di un trigger di funzione Lambda

È necessario disporre dell'ARN di una AWS Lambda funzione personalizzata prima di poter generare una regola personalizzata AWS Config . Segui queste istruzioni per creare una funzione di base Node.js che restituisca un valore conforme o non conforme in AWS Config base allo stato della risorsa. `XrayEncryptionConfig`

Per creare una funzione Lambda con un trigger di modifica AWS::XrayEncryptionConfig

1. Aprire la console Lambda. Scegli Crea funzione.

2. Scegliere Blueprints (Progetti) e quindi filtrare la libreria dei progetti in base al progetto config-rule-change-triggered. Fare clic sul collegamento nel nome del progetto o scegliere Configure (Configura) per proseguire.
3. Definire i seguenti campi per configurare il progetto:
   - Digitare un nome nel campo Name (Nome).
   - In Role (Ruolo), seleziona Create new role from template(s) (Crea nuovo ruolo da modello/i):
   - In Role Name (Nome ruolo) digita un nome.
   - In Policy templates (Modelli di policy), scegliere AWS Config Rules permissions (Regole di autorizzazione).
4. Scegli Crea funzione per creare e visualizzare la tua funzione nella AWS Lambda console.
5. Modificare il codice della funzione per sostituire `AWS::EC2::Instance` con `AWS::XrayEncryptionConfig`. Puoi anche aggiornare il campo descrizione affinché rifletta questa modifica.

   Codice di default

```
    if (configurationItem.resourceType !== 'AWS::EC2::Instance') {
        return 'NOT_APPLICABLE';
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
        return 'NON_COMPLIANT';
```

   Codice aggiornato

```
    if (configurationItem.resourceType !== 'AWS::XRay::EncryptionConfig') {
        return 'NOT_APPLICABLE';
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
        return 'NON_COMPLIANT';
```

6. Aggiungi quanto segue al tuo ruolo di esecuzione in IAM per accedere a X-Ray. Queste autorizzazioni consentono l'accesso in sola lettura alle tue risorse. X-Ray La mancata fornitura

dell'accesso alle risorse appropriate genererà un messaggio fuori ambito nel momento in AWS Config cui si valuta la funzione Lambda associata alla regola.

```
{
    "Sid": "Stmt1529350291539",
    "Action": [
        "xray:GetEncryptionConfig"
    ],
    "Effect": "Allow",
    "Resource": "*"
 }
```

Creazione di una personalizzazione AWS Config regola per la radiografia

Quando viene creata la funzione Lambda, annota l'ARN della funzione e vai alla AWS Config console per creare la tua regola personalizzata.

Per creare un AWS Config regola per X-Ray

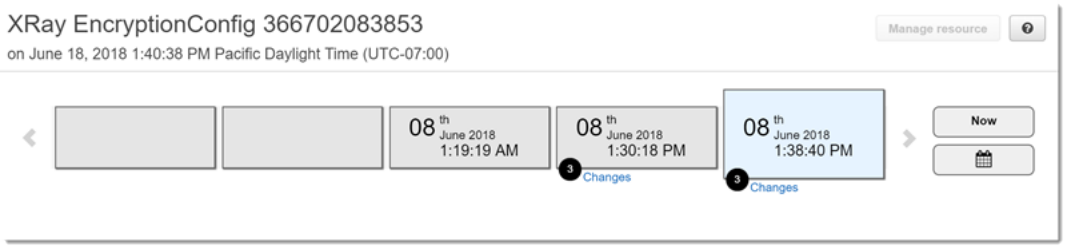
1. Aprire la pagina Rules (Regole) della console di AWS Config
2. Scegliere Add rule (Aggiungi regola) e quindi scegliere Add custom rule (Aggiungi regola personalizzata).
3. In AWS Lambda Function ARN, inserisci l'ARN associato alla funzione Lambda che desideri utilizzare.
4. Scegliere il tipo di trigger da impostare:
   - Modifiche alla configurazione: AWS Config attiva la valutazione quando una risorsa che corrisponde all'ambito della regola cambia nella configurazione. La valutazione viene eseguita dopo l' AWS Config invio di una notifica di modifica degli elementi di configurazione.
   - Periodico: AWS Config esegue le valutazioni della regola con una frequenza scelta (ad esempio, ogni 24 ore).
5. In Resource type (Tipo risorsa), scegliere EncryptionConfig nella sezione X-Ray .
6. Selezionare Salva.

La AWS Config console inizia immediatamente a valutare la conformità della regola. La valutazione può richiedere alcuni minuti.

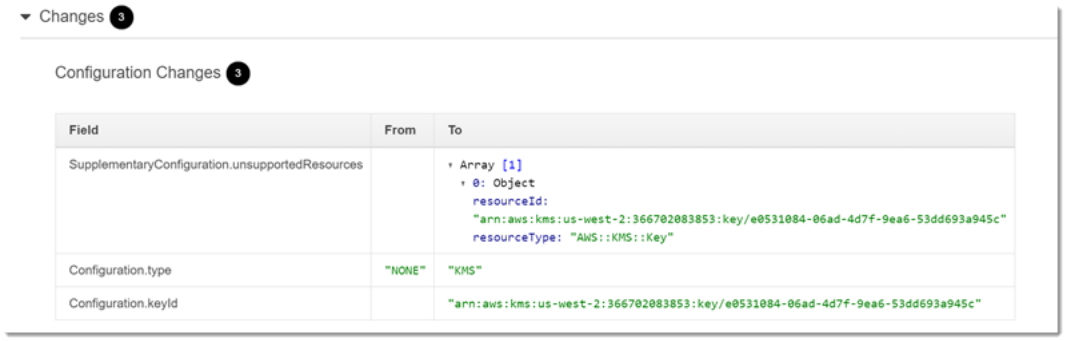
Ora che questa regola è conforme, AWS Config puoi iniziare a compilare una cronologia degli audit. AWS Config registra le modifiche alle risorse sotto forma di cronologia. Per ogni modifica nella timeline degli eventi, AWS Config genera una tabella in un from/to formato per mostrare cosa è cambiato nella rappresentazione JSON della chiave di crittografia. Le due modifiche ai campi associate EncryptionConfig sono `Configuration.type` e. `Configuration.keyID`

Risultati di esempio

Di seguito è riportato un esempio di AWS Config cronologia che mostra le modifiche apportate in date e ore specifiche.



Di seguito è riportato un esempio di AWS Config modifica. Il from/to formato illustra cosa è cambiato. Questo esempio mostra che le impostazioni di X-Ray crittografia predefinite sono state modificate in una chiave di crittografia definita.



Notifiche Amazon SNS

Per ricevere notifiche sulle modifiche AWS Config alla configurazione, imposta la pubblicazione delle notifiche di Amazon SNS. Per ulteriori informazioni, consulta Monitoraggio AWS Config delle modifiche alle risorse tramite e-mail.

AWS AppSync e AWS X-Ray

È possibile abilitare e tracciare le richieste per. AWS AppSync Per ulteriori informazioni, vedere Tracing with AWS X-Ray per istruzioni.

Quando il tracciamento X-Ray è abilitato per un' AWS AppSync API, nell'account viene creato automaticamente un ruolo collegato al servizio AWS Identity and Access Management con le autorizzazioni appropriate. Ciò consente di AWS AppSync inviare tracce a X-Ray in modo sicuro.

Supporto di tracciamento attivo di Amazon API Gateway per AWS X-Ray

Puoi usare X-Ray per tracciare e analizzare le richieste degli utenti mentre viaggiano attraverso il tuo Amazon API Gateway APIs verso i servizi sottostanti. API Gateway supporta il tracciamento a raggi X per tutti i tipi di endpoint API Gateway: regionali, ottimizzati per l'edge e privati. Puoi usare X-Ray con Amazon API Gateway ovunque Regioni AWS X-Ray sia disponibile. Per ulteriori informazioni, consulta Trace API Gateway API Execution with AWS X-Ray nella Amazon API Gateway Developer Guide.

> **Note**
>
> X-Ray supporta solo il tracciamento per REST tramite API APIs Gateway.

Amazon API Gateway fornisce supporto di tracciamento attivo per AWS X-Ray. Abilita il tracciamento attivo sulle fasi dell'API per campionare le richieste in arrivo e inviare tracce a X-Ray.

Per abilitare il tracciamento attivo su una fase API

1. Apri la console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway/.
2. Scegliere un'API.
3. Scegliere una fase.
4. Nella scheda Log/Tracciamento, scegli Abilita tracciamento X-Ray, quindi scegli Salva modifiche.
5. Scegliere Resources (Risorse) sul pannello di navigazione a sinistra.
6. Per ridistribuire l'API con le nuove impostazioni, scegli il menu a discesa Azioni, quindi scegli Deploy API.

API Gateway utilizza le regole di campionamento definite nella console X-Ray per determinare quali richieste registrare. È possibile creare regole che si applicano solo a APIs, o che si applicano solo alle richieste che contengono determinate intestazioni. API Gateway registra le intestazioni

negli attributi del segmento, insieme ai dettagli sulla fase e sulla richiesta. Per ulteriori informazioni, consulta Configurazione delle regole di campionamento di.

> **Note**
>
> Quando si traccia REST APIs con l'integrazione HTTP di API Gateway, il nome di servizio di ogni segmento viene impostato sul percorso dell'URL della richiesta da API Gateway all'endpoint di integrazione HTTP, ottenendo un nodo di servizio sulla mappa di tracciamento a raggi X per ogni percorso URL univoco. Un numero elevato di percorsi URL può far sì che la mappa di traccia superi il limite di 10.000 nodi, con conseguente errore.
> Per ridurre al minimo il numero di nodi di servizio creati da API Gateway, valuta la possibilità di passare i parametri all'interno della stringa di query dell'URL o nel corpo della richiesta tramite POST. Entrambi gli approcci garantiranno che i parametri non facciano parte del percorso URL, il che potrebbe comportare un minor numero di percorsi URL e nodi di servizio distinti.

Per tutte le richieste in entrata, API Gateway aggiunge un'intestazione di tracciamento alle richieste HTTP in entrata che non ne hanno già una.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Formato ID di traccia X-Ray

Una radiografia `trace_id` è composta da tre numeri separati da trattini. Ad esempio `1-58406520-a006649127e371903a2de979`. Questo include:

- Il numero di versione, che è. 1
- L'ora della richiesta originale in Unix epoch time utilizzando 8 cifre esadecimali.

  Ad esempio, il fuso orario PST delle 10:00 del 1° dicembre 2016 è espresso in secondi o in cifre esadecimali. `1480615200` `58406520`
- Un identificatore a 96 bit univoco a livello globale per la traccia in 24 cifre esadecimali.

Se il tracciamento attivo è disattivato, la fase memorizza comunque un segmento se la richiesta proviene da un servizio che ha campionato la richiesta e ha avviato un tracciamento. Ad esempio, un'applicazione Web strumentata può chiamare un'API API Gateway con un client HTTP. Quando si

strumenta un client HTTP con l'SDK X-Ray, aggiunge un'intestazione di tracciamento alla richiesta in uscita che contiene la decisione di campionamento. API Gateway legge l'intestazione di tracciamento e crea un segmento per le richieste campionate.

Se utilizzi API Gateway per generare un SDK Java per la tua API, puoi strumentare il client SDK aggiungendo un gestore di richieste con il client builder, nello stesso modo in cui strumenteresti manualmente un client SDK. AWS Per istruzioni, consulta Tracciamento delle chiamate AWS SDK con X-Ray SDK for Java.

Amazon EC2 e AWS App Mesh

AWS X-Ray si integra con la gestione dei proxy AWS App Mesh Envoy per microservizi. App Mesh fornisce una versione di Envoy che puoi configurare per inviare dati di traccia al X-Ray demone in esecuzione in un contenitore della stessa attività o pod. X-Ray supporta il tracciamento con i seguenti servizi compatibili con App Mesh:

- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Amazon Elastic Compute Cloud (Amazon EC2)

Usa le seguenti istruzioni per scoprire come abilitare il X-Ray tracciamento tramite App Mesh.



Per configurare il proxy Envoy a cui inviare dati X-Ray, imposta la variabile di `ENABLE_ENVOY_XRAY_TRACING` ambiente nella definizione del contenitore.

> **Note**
>
> La versione App Mesh di Envoy attualmente non invia tracce in base alle regole di campionamento configurate. https://docs.aws.amazon.com/xray/latest/devguide/xray-console-sampling.html Utilizza invece una frequenza di campionamento fissa del 5% per la versione

1.16.3 o successiva di Envoy o una frequenza di campionamento del 50% per le versioni di Envoy precedenti alla 1.16.3.

Example Definizione del contenitore Envoy per 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

Per ulteriori informazioni sugli indirizzi delle regioni Envoy disponibili, consulta l'immagine di Envoy nella Guida per l'utente. AWS App Mesh

Per dettagli sull'esecuzione del X-Ray demone in un contenitore, consulta. Esecuzione del daemon X-Ray su Amazon ECS Per un'applicazione di esempio che include una service mesh, un microservizio, un proxy Envoy e un X-Ray daemon, distribuisci l'`colorapp`esempio nel repository App Mesh Examples. GitHub

Ulteriori informazioni

- Nozioni di base su AWS App Mesh
- Guida introduttiva ad Amazon ECS AWS App Mesh

AWS App Runner e X-Ray

AWS App Runner offre un Servizio AWS modo rapido, semplice ed economico per la distribuzione dal codice sorgente o da un'immagine del contenitore direttamente a un'applicazione web scalabile e sicura in. Cloud AWS Non è necessario apprendere nuove tecnologie, decidere quale servizio di elaborazione utilizzare o sapere come fornire e configurare le risorse. AWS Vedi Cos'è AWS App Runner per ulteriori informazioni.

AWS App Runner invia tracce a X-Ray integrandosi con Distro OpenTelemetry for (AWS ADOT). Usa ADOT per SDKs raccogliere dati di traccia per le tue applicazioni containerizzate e usa X-Ray per analizzare e ottenere informazioni dettagliate sulla tua applicazione strumentata. Per ulteriori informazioni, vedere Tracciamento per l'applicazione App Runner con X-Ray.

Registrazione delle chiamate all'API X-Ray con AWS CloudTrail

AWS X-Ray è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, ruolo o un Servizio AWS. CloudTrail acquisisce tutte le chiamate API per X-Ray come eventi. Le chiamate acquisite includono chiamate dalla console X-Ray e chiamate in codice alle operazioni dell'API X-Ray. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a X-Ray, l'indirizzo IP da cui è stata effettuata la richiesta, quando è stata effettuata e ulteriori dettagli.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali utente root o utente.
- Se la richiesta è stata effettuata per conto di un utente del Centro identità IAM.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro Servizio AWS.

CloudTrail è attivo nel tuo account Account AWS quando crei l'account e hai automaticamente accesso alla cronologia degli CloudTrail eventi. La cronologia CloudTrail degli eventi fornisce un record visualizzabile, ricercabile, scaricabile e immutabile degli ultimi 90 giorni di eventi di gestione registrati in un. Regione AWSPer ulteriori informazioni, consulta Lavorare con la cronologia degli CloudTrail eventi nella Guida per l'utente.AWS CloudTrail Non sono CloudTrail previsti costi per la visualizzazione della cronologia degli eventi.

Per una registrazione continua degli eventi degli Account AWS ultimi 90 giorni, crea un trail o un data store di eventi CloudTrailLake.

CloudTrail sentieri

Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Tutti i percorsi creati utilizzando il Console di gestione AWS sono multiregionali. È possibile creare un trail per una singola Regione o per più Regioni tramite AWS CLI. La creazione di un percorso multiregionale è consigliata in quanto consente di registrare l'intera attività del proprio Regioni AWS account. Se si crea un trail per una singola Regione, è possibile visualizzare solo gli eventi registrati nella Regione AWS del trail. Per ulteriori informazioni sui trail, consulta Creating a trail for your Account AWS e Creating a trail for an organization nella Guida per l'utente di AWS CloudTrail .

Puoi inviare gratuitamente una copia dei tuoi eventi di gestione in corso al tuo bucket Amazon S3 CloudTrail creando un percorso, tuttavia ci sono costi di storage di Amazon S3. Per ulteriori informazioni sui CloudTrail prezzi, consulta la pagina Prezzi.AWS CloudTrail Per informazioni sui prezzi di Amazon S3, consulta Prezzi di Amazon S3.

CloudTrail Lake Event Data Store

CloudTrail Lake ti consente di eseguire query basate su SQL sui tuoi eventi. CloudTrail Lake converte gli eventi esistenti in formato JSON basato su righe in formato Apache ORC. ORC è un formato di archiviazione a colonne ottimizzato per il recupero rapido dei dati. Gli eventi vengono aggregati in archivi di dati degli eventi, che sono raccolte di eventi immutabili basate sui criteri selezionati applicando i selettori di eventi avanzati. I selettori applicati a un archivio di dati degli eventi controllano quali eventi persistono e sono disponibili per l'esecuzione della query. Per ulteriori informazioni su CloudTrail Lake, consulta Working with AWS CloudTrail Lake nella Guida per l'utente.AWS CloudTrail

CloudTrail Gli archivi e le richieste di dati sugli eventi di Lake comportano dei costi. Quando crei un datastore di eventi, scegli l'opzione di prezzo da utilizzare per tale datastore. L'opzione di prezzo determina il costo per l'importazione e l'archiviazione degli eventi, nonché il periodo di

conservazione predefinito e quello massimo per il datastore di eventi. Per ulteriori informazioni sui CloudTrail prezzi, consulta Prezzi.AWS CloudTrail

Argomenti

- Eventi di gestione dei raggi X in CloudTrail
- Eventi relativi ai dati a raggi X in CloudTrail
- Esempi di eventi X-Ray

Eventi di gestione dei raggi X in CloudTrail

AWS X-Ray si integra con AWS CloudTrail per registrare le azioni API eseguite da un utente, un ruolo o un utente Servizio AWS in X-Ray. Puoi utilizzarlo CloudTrail per monitorare le richieste API X-Ray in tempo reale e archiviare i log in Amazon S3, Amazon Logs e Amazon Events. CloudWatch CloudWatch X-Ray supporta la registrazione delle seguenti azioni come eventi nei CloudTrail file di registro:

Operazioni API supportate

- PutEncryptionConfig
- GetEncryptionConfig
- CreateGroup
- UpdateGroup
- DeleteGroup
- GetGroup
- GetGroups
- GetInsight
- GetInsightEvents
- GetInsightImpactGraph
- GetInsightSummaries
- GetSamplingStatisticSummaries

Eventi relativi ai dati a raggi X in CloudTrail

Gli eventi di dati forniscono informazioni sulle operazioni sulle risorse eseguite su o in una risorsa (ad esempio PutTraceSegments, che carica i documenti dei segmenti su X-Ray).

Queste operazioni sono definite anche operazioni del piano dei dati. Gli eventi di dati sono spesso attività che interessano volumi elevati di dati. Per impostazione predefinita, CloudTrail non registra gli eventi relativi ai dati. La cronologia CloudTrail degli eventi non registra gli eventi relativi ai dati.

Per gli eventi di dati sono previsti costi aggiuntivi. Per ulteriori informazioni sui CloudTrail prezzi, consulta la sezione AWS CloudTrail Prezzi.

È possibile registrare gli eventi relativi ai dati per i tipi di risorse X-Ray utilizzando la CloudTrail console o AWS CLI le operazioni CloudTrail API. Per ulteriori informazioni su come registrare gli eventi relativi ai dati, consulta Registrazione di eventi relativi ai dati con Console di gestione AWS e Registrazione di eventi di dati con AWS Command Line Interface nella Guida per l'utente AWS CloudTrail .

La tabella seguente elenca i tipi di risorse X-Ray per i quali è possibile registrare gli eventi relativi ai dati. La colonna Data event type (console) mostra il valore da scegliere dall'elenco Data event type (console) sulla CloudTrail console. La colonna del valore resources.type mostra il `resources.type` valore da specificare durante la configurazione dei selettori di eventi avanzati utilizzando o. AWS CLI CloudTrail APIs La CloudTrail colonna Dati APIs registrati mostra le chiamate API registrate per il tipo di risorsa. CloudTrail

Tipo di evento di dati (console)	valore resources.type	Dati registrati APIs su CloudTrail
Traccia a raggi X	`AWS::XRay::Trace`	• PutTraceSegments • GetTraceSummaries • GetTraceGraph • GetServiceGraph • BatchGetTraces • GetTimeSeriesServiceStatistics • PutTelemetryRecords • GetSamplingTargets

Puoi configurare selettori di eventi avanzati per filtrare `readOnly` i campi `eventName` e registrare solo gli eventi che ritieni importanti. Tuttavia, non è possibile selezionare gli eventi aggiungendo il selettore di `resources.ARN` campo, poiché le tracce X-Ray non lo sono. ARNs Per maggiori informazioni su questi campi, consulta AdvancedFieldSelector nella Guida di riferimento delle API di AWS CloudTrail . Di seguito è riportato un esempio di come eseguire il `put-event-selectors` AWS CLI comando per registrare gli eventi relativi ai dati su un CloudTrail percorso. È necessario eseguire il comando o specificare la regione in cui è stato creato il trail; in caso contrario, l'operazione restituisce un'`InvalidHomeRegionException`eccezione.

```
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"
      }
   ]
}'
```

Esempi di eventi X-Ray

Esempio di evento gestionale, **`GetEncryptionConfig`**

Di seguito è riportato un esempio di registrazione del GetEncryptionConfig registro X-Ray. CloudTrail

Example

```
{
    "eventVersion"=>"1.05",
    "userIdentity"=>{
        "type"=>"AssumedRole",
        "principalId"=>"AROAJVHBZWD3DN6CI2MHM:MyName",
        "arn"=>"arn:aws:sts::123456789012:assumed-role/MyRole/MyName",
        "accountId"=>"123456789012",
        "accessKeyId"=>"AKIAIOSFODNN7EXAMPLE",
        "sessionContext"=>{
            "attributes"=>{
```

```
                "mfaAuthenticated"=>"false",
                "creationDate"=>"2023-7-01T00:24:36Z"
            },
            "sessionIssuer"=>{
                "type"=>"Role",
                "principalId"=>"AROAJVHBZWD3DN6CI2MHM",
                "arn"=>"arn:aws:iam::123456789012:role/MyRole",
                "accountId"=>"123456789012",
                "userName"=>"MyRole"
            }
        }
    },
    "eventTime"=>"2023-7-01T00:24:36Z",
    "eventSource"=>"xray.amazonaws.com",
    "eventName"=>"GetEncryptionConfig",
    "awsRegion"=>"us-east-2",
    "sourceIPAddress"=>"33.255.33.255",
    "userAgent"=>"aws-sdk-ruby2/2.11.19 ruby/2.3.1 x86_64-linux",
    "requestParameters"=>nil,
    "responseElements"=>nil,
    "requestID"=>"3fda699a-32e7-4c20-37af-edc2be5acbdb",
    "eventID"=>"039c3d45-6baa-11e3-2f3e-e5a036343c9f",
    "eventType"=>"AwsApiCall",
    "recipientAccountId"=>"123456789012"
}
```

Esempio di evento relativo ai dati, **`PutTraceSegments`**

Di seguito è riportato un esempio di immissione nel registro degli eventi PutTraceSegments dei dati X-Ray. CloudTrail

Example

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAWYXPW54Y4NEXAMPLE:i-0dzz2ac111c83zz0z",
    "arn": "arn:aws:sts::012345678910:assumed-role/my-service-role/
i-0dzz2ac111c83zz0z",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
```

```
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAWYXPW54Y4NEXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/service-role/my-service-role",
        "accountId": "012345678910",
        "userName": "my-service-role"
      },
      "attributes": {
        "creationDate": "2024-01-22T17:34:11Z",
        "mfaAuthenticated": "false"
      },
      "ec2RoleDelivery": "2.0"
    }
  },
  "eventTime": "2024-01-22T18:22:05Z",
  "eventSource": "xray.amazonaws.com",
  "eventName": "PutTraceSegments",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.0",
  "userAgent": "aws-sdk-ruby3/3.190.0 md/internal ua/2.0 api/xray#1.0.0 os/linux md/
x86_64 lang/ruby#2.7.8 md/2.7.8 cfg/retry-mode#legacy",
  "requestParameters": {
    "traceSegmentDocuments": [
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0000",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0000",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0001",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0002"
    ]
  },
  "responseElements": {
    "unprocessedTraceSegments": []
  },
  "requestID": "5zzzzz64-acbd-46ff-z544-451a3ebcb2f8",
  "eventID": "4zz51z7z-77f9-44zz-9bd7-6c8327740f2e",
  "readOnly": false,
  "resources": [
    {
      "type": "AWS::XRay::Trace"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "012345678910",
  "eventCategory": "Data",
```

```
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ZZZZZ-RSA-AAA128-GCM-SHA256",
    "clientProvidedHostHeader": "example.us-west-2.xray.cloudwatch.aws.dev"
  }
}
```

CloudWatch integrazione con X-Ray

AWS X-Ray si integra con CloudWatch Application Signals, CloudWatch RUM e CloudWatch Synthetics per semplificare il monitoraggio dello stato delle applicazioni. Consenti alla tua applicazione Application Signals di monitorare e risolvere i problemi operativi dei tuoi servizi, delle pagine client, dei canali Synthetics e delle dipendenze dei servizi.

Correlando CloudWatch metriche, log e tracce a raggi X, la mappa di tracciamento a raggi X fornisce una end-to-end panoramica dei tuoi servizi per aiutarti a individuare rapidamente i punti deboli delle prestazioni e identificare gli utenti interessati.

Con CloudWatch RUM, è possibile eseguire il monitoraggio reale degli utenti per raccogliere e visualizzare dati lato client sulle prestazioni delle applicazioni Web dalle sessioni utente effettive in tempo quasi reale. Con AWS X-Ray and CloudWatch RUM, è possibile analizzare ed eseguire il debug del percorso della richiesta a partire dagli utenti finali dell'applicazione tramite servizi gestiti a valle. AWS In questo modo, puoi identificare le tendenze e gli errori di latenza che hanno un impatto sugli utenti finali.

Argomenti

- CloudWatch RUM e AWS X-Ray
- Eseguire il debug di canarini CloudWatch sintetici utilizzando X-Ray

CloudWatch RUM e AWS X-Ray

Con Amazon CloudWatch RUM, puoi eseguire il monitoraggio reale degli utenti per raccogliere e visualizzare dati lato client sulle prestazioni delle tue applicazioni Web da sessioni utente effettive in tempo quasi reale. Con AWS X-Ray and CloudWatch RUM, puoi analizzare ed eseguire il debug del percorso della richiesta partendo dagli utenti finali della tua applicazione attraverso servizi gestiti a valle. AWS In questo modo, puoi identificare le tendenze e gli errori di latenza che hanno un impatto sugli utenti finali.

Dopo aver attivato il tracciamento a raggi X delle sessioni utente, CloudWatch RUM aggiunge un'intestazione di traccia a raggi X alle richieste HTTP consentite e registra un segmento a raggi X per le richieste HTTP consentite. È quindi possibile visualizzare le tracce e i segmenti di queste sessioni utente in X-Ray CloudWatch e nelle console, inclusa la mappa di tracciamento X-Ray.

> **Note**
>
> CloudWatch RUM non si integra con le regole di campionamento a raggi X. Scegliete invece una percentuale di campionamento quando configurate l'applicazione per utilizzare RUM. CloudWatch Le tracce inviate dal CloudWatch RUM potrebbero comportare costi aggiuntivi. Per ulteriori informazioni, consulta Prezzi di AWS X-Ray.

Per impostazione predefinita, le tracce lato client inviate da CloudWatch RUM non sono connesse alle tracce lato server. Per connettere le tracce lato client con le tracce lato server, configura il client web CloudWatch RUM per aggiungere un'intestazione di traccia X-Ray a queste richieste HTTP.

> **Warning**
>
> La configurazione del client web CloudWatch RUM per aggiungere un'intestazione di traccia X-Ray alle richieste HTTP può causare il fallimento della condivisione delle risorse tra le origini (CORS). Per evitare ciò, aggiungete l'intestazione `X-Amzn-Trace-Id` HTTP all'elenco delle intestazioni consentite nella configurazione CORS del servizio downstream. Se utilizzi API Gateway come downstream, consulta Abilitazione di CORS per una risorsa API REST. Si consiglia vivamente di testare l'applicazione prima di aggiungere un'intestazione di traccia X-Ray lato client in un ambiente di produzione. Per ulteriori informazioni, consultate la documentazione del client web CloudWatch RUM.

Per ulteriori informazioni sul monitoraggio degli utenti reali in CloudWatch, consulta Use CloudWatch RUM. Per configurare l'applicazione per l'utilizzo di CloudWatch RUM, incluso il tracciamento delle sessioni utente con X-Ray, consulta Configurare un'applicazione per CloudWatch utilizzare RUM.

Eseguire il debug di canarini CloudWatch sintetici utilizzando X-Ray

CloudWatch Synthetics è un servizio completamente gestito che consente di monitorare gli endpoint APIs e di utilizzare canari con script che funzionano 24 ore al giorno, una volta al minuto.

È possibile personalizzare gli script Canary per verificare le modifiche in:

- Disponibilità
- Latenza
- Transazioni
- Link interrotti o morti
- Step-by-step completamenti delle attività
- Errori di caricamento della pagina
- Latenze di carico per le risorse dell'interfaccia utente
- Flussi della procedura guidata complessi
- Flussi di checkout nella tua applicazione

I Canary seguono gli stessi percorsi, eseguono le stesse azioni e comportamenti dei tuoi clienti e verificano continuamente l'esperienza del cliente.

Per ulteriori informazioni sulla configurazione dei test, consulta Utilizzo di Synthetics per creare e gestire i Canary.



Gli esempi seguenti mostrano casi d'uso comuni per problemi di debug che i Canary di Synthetics generano. Ogni esempio dimostra una strategia chiave per il debug utilizzando la mappa di traccia o la console X-Ray Analytics.

Per ulteriori informazioni su come leggere e interagire con la mappa di traccia, vedere Visualizzazione della mappa dei servizi.

Per ulteriori informazioni su come leggere e interagire con la console di X-Ray Analytics, consulta Interagire con la AWS X-Ray console Analytics.

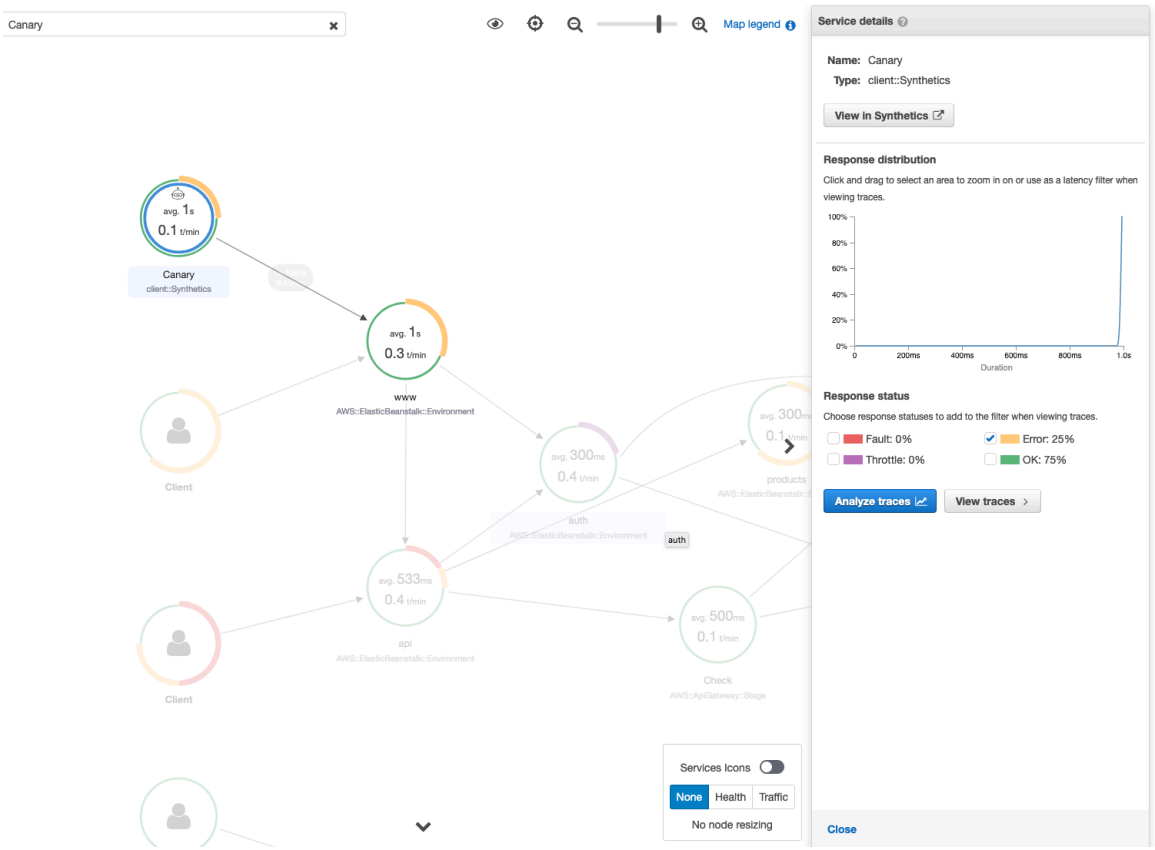
Argomenti

- Visualizza i canarini con una maggiore segnalazione degli errori nella mappa di tracciamento
- Utilizza le mappe dei dettagli di traccia per le singole tracce per visualizzare ogni richiesta in dettaglio
- Determinare la causa principale degli errori in corso nei servizi upstream e downstream
- Identificare i colli di bottiglia e le tendenze delle prestazioni
- Confrontare latenza ed errori o tassi di errore prima e dopo le modifiche
- Determina la copertura canarina richiesta per tutti e APIs URLs
- Utilizzare i gruppi per concentrarsi sui test Synthetics.

Visualizza i canarini con una maggiore segnalazione degli errori nella mappa di tracciamento

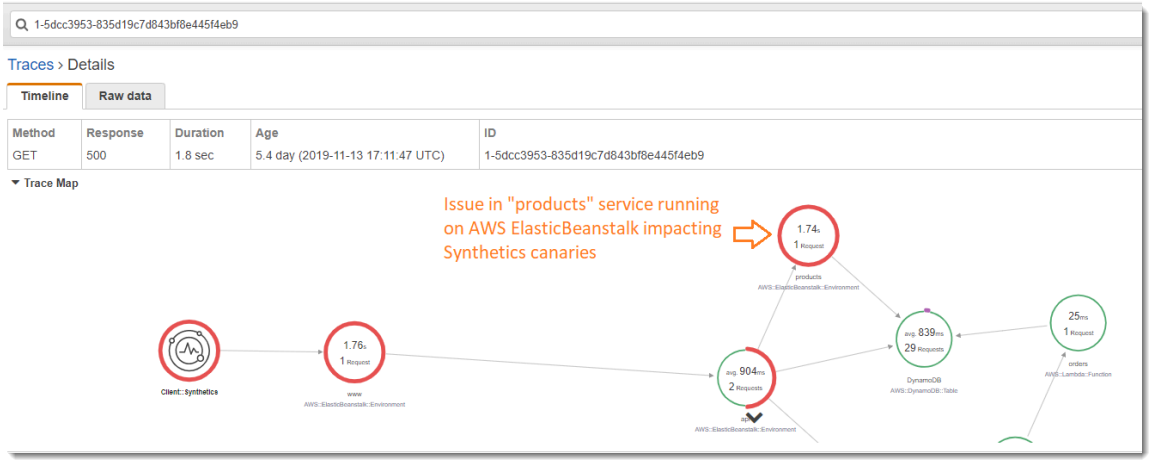
Per vedere quali canarini presentano un aumento di errori, guasti, velocità di limitazione o tempi di risposta lenti all'interno della mappa di tracciamento X-Ray, puoi evidenziare i nodi client Synthetics Canary utilizzando il filtro. `Client::Synthetic` Facendo clic su un nodo viene visualizzata la distribuzione del tempo di risposta dell'intera richiesta. Facendo clic su un bordo tra due nodi vengono visualizzati i dettagli sulle richieste che hanno percorso quella connessione. È inoltre possibile visualizzare i nodi dedotti «remoti» per i servizi downstream correlati nella mappa di traccia.

Quando fai clic sul nodo Synthetics, c'è un pulsante Visualizza in Synthetics sul pannello laterale che ti reindirizza alla console Synthetics dove puoi controllare i dettagli del canarino.



Utilizza le mappe dei dettagli di traccia per le singole tracce per visualizzare ogni richiesta in dettaglio

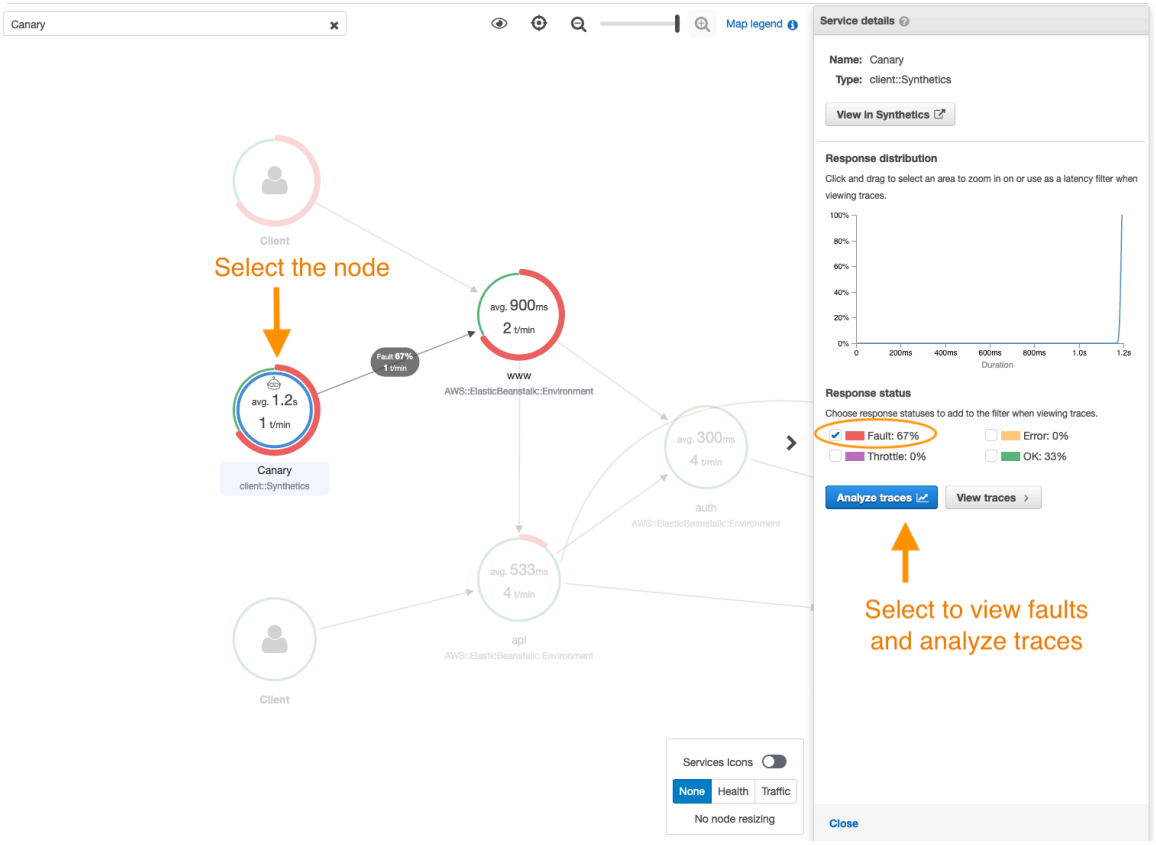
Per determinare quale servizio genera la maggiore latenza o causa un errore, richiama la mappa dei dettagli di traccia selezionando la traccia nella mappa di traccia. Le singole mappe dei dettagli di traccia mostrano il end-to-end percorso di una singola richiesta. Utilizza questa opzione per comprendere i servizi richiamati e visualizzare i servizi upstream e downstream.

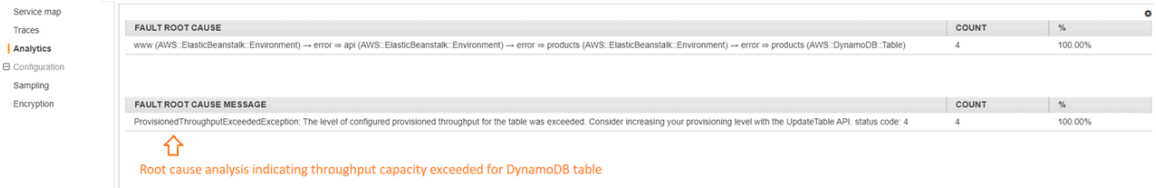


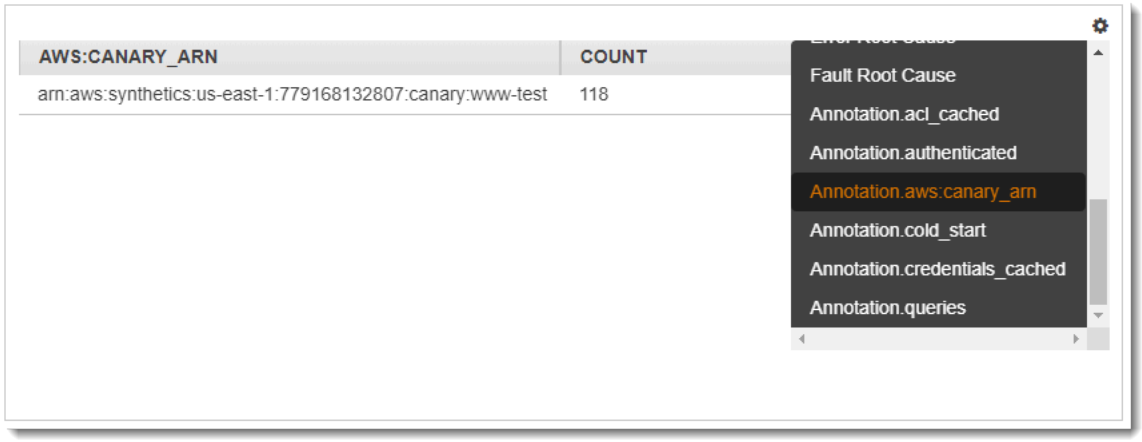
Determinare la causa principale degli errori in corso nei servizi upstream e downstream

Una volta ricevuto un CloudWatch allarme per guasti in un sistema Synthetics Canary, utilizzate la modellazione statistica sui dati di traccia in X-Ray per determinare la probabile causa principale del problema all'interno della console X-Ray Analytics. Nella console Analytics, la tabella Response Time Root Cause mostra i percorsi delle entità registrate. X-Ray determina quale percorso della traccia è la causa più probabile del tempo di risposta. Il formato indica una gerarchia di entità incontrate che termina in una causa principale del tempo di risposta.

L'esempio seguente mostra che il test Synthetics per l'API «XXX» in esecuzione su API Gateway non riesce a causa di un'eccezione di capacità di throughput nella tabella Amazon DynamoDB.

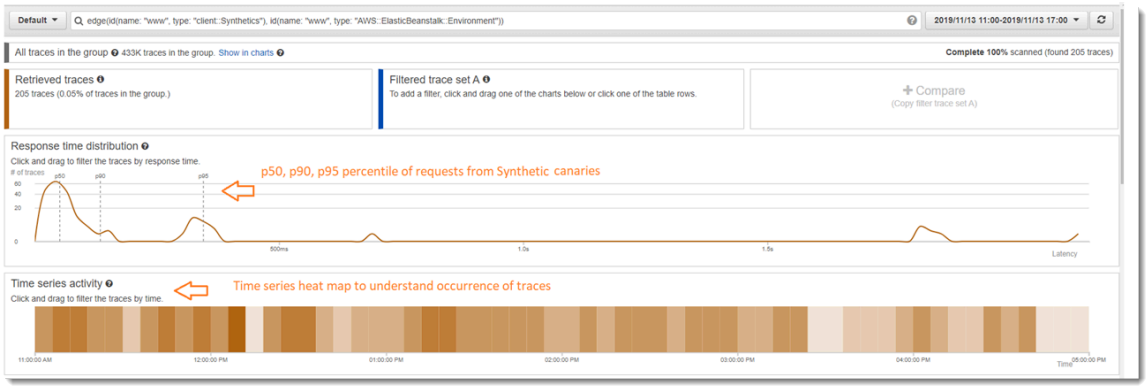






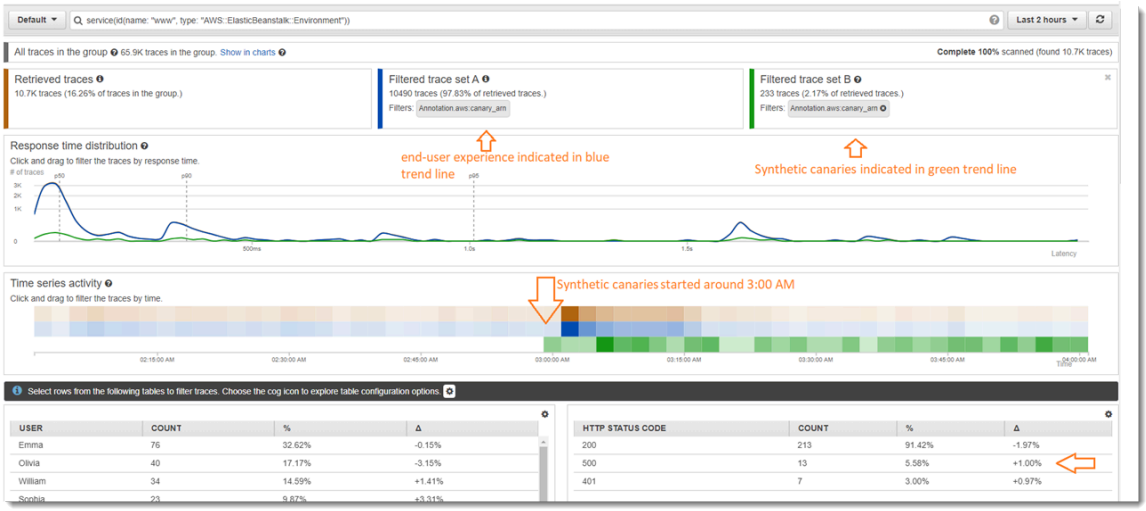
Identificare i colli di bottiglia e le tendenze delle prestazioni

Puoi visualizzare le tendenze delle prestazioni del tuo endpoint nel tempo utilizzando il traffico continuo proveniente dai tuoi canali Synthetics per compilare una mappa dei dettagli di tracciamento per un periodo di tempo.



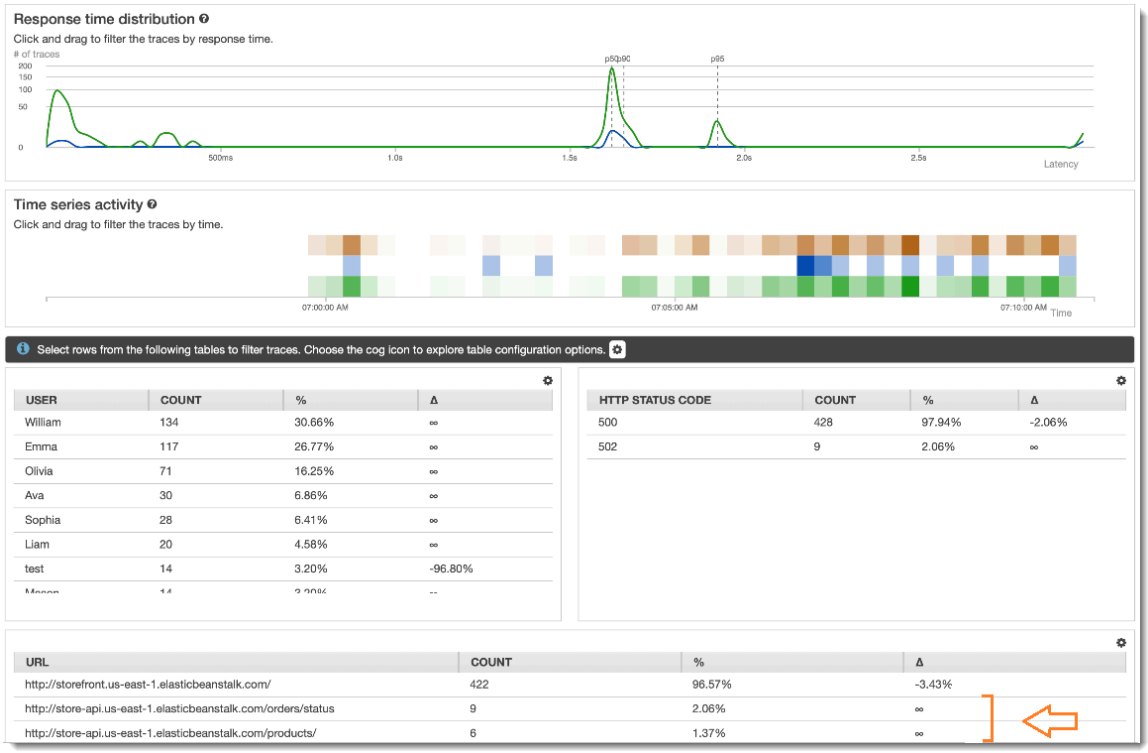
Confrontare latenza ed errori o tassi di errore prima e dopo le modifiche

Pinpoint l'ora in cui si è verificata una modifica per correlarla all'aumento dei problemi riscontrati dai vostri canarini. Usa la console X-Ray Analytics per definire gli intervalli di tempo prima e dopo come set di tracce diversi, creando una differenziazione visiva nella distribuzione dei tempi di risposta.



Determina la copertura canarina richiesta per tutti e APIs URLs

Usa X-Ray Analytics per confrontare l'esperienza dei Canary con gli utenti. L'interfaccia utente seguente mostra una linea di tendenza blu per i Canary e una linea verde per gli utenti. Puoi anche identificare che due su tre URLs non hanno il test canarino.



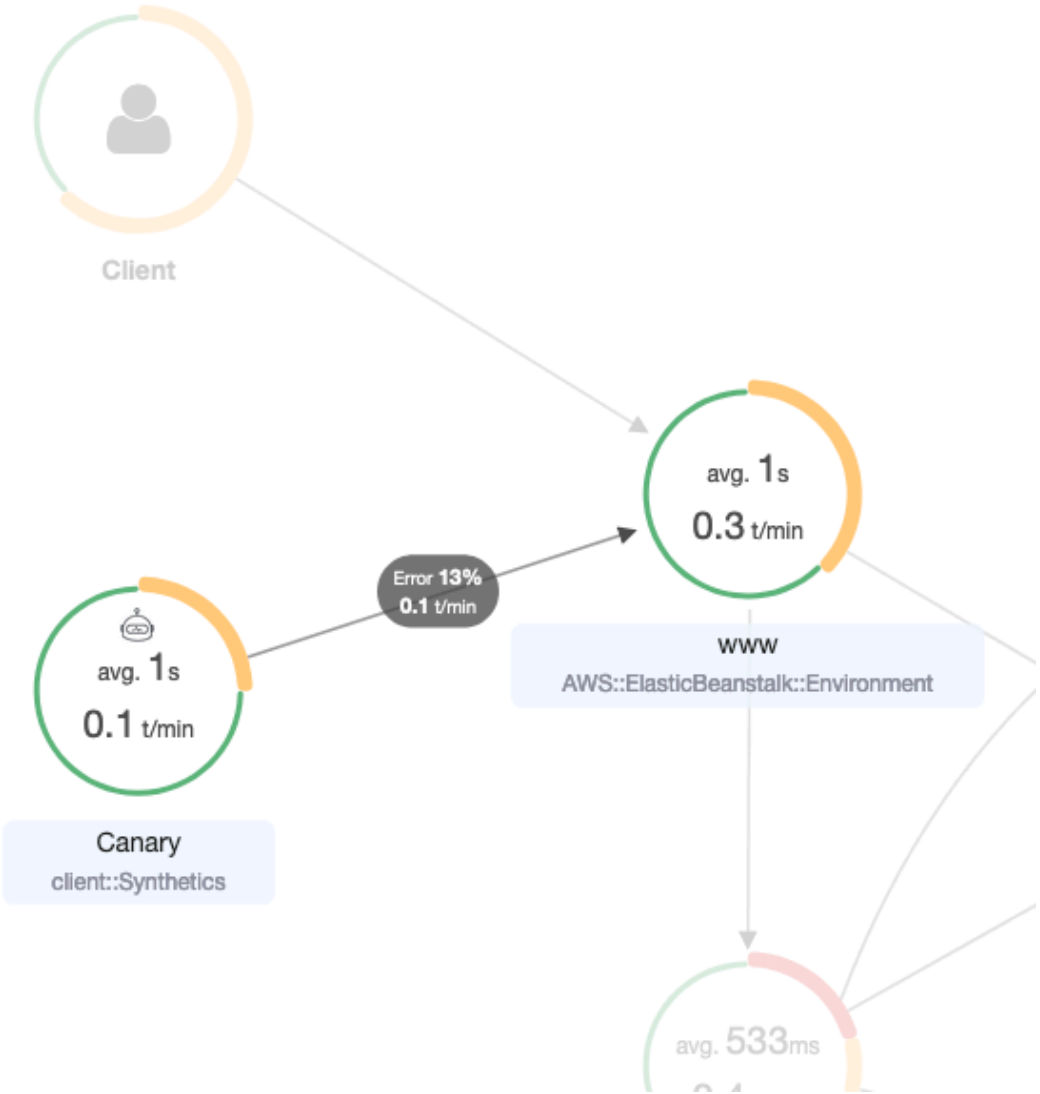
Utilizzare i gruppi per concentrarsi sui test Synthetics.

È possibile creare un gruppo X-Ray utilizzando un'espressione di filtro per concentrarsi su un determinato insieme di flussi di lavoro, ad esempio i test Synthetics per l'applicazione «www» in

esecuzione. AWS Elastic Beanstalk Utilizzate le parole chiave complesse e `service()` filtrate attraverso servizi e `edge()` bordi.

Example Espressione filtro gruppo

```
"edge(id(name: "www", type: "client::Synthetics"), id(name: "www", type:
 "AWS::ElasticBeanstalk::Environment"))"
```



AWS Elastic Beanstalk e AWS X-Ray

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

AWS Elastic Beanstalk le piattaforme includono il demone X-Ray. È possibile eseguire il demone impostando un'opzione nella console Elastic Beanstalk o con un file di configurazione.

Nella piattaforma Java SE, puoi utilizzare un file Buildfile per creare un'applicazione con Maven o Gradle sull'istanza. Gli X-Ray SDK for Java AWS SDK per Java sono disponibili presso Maven, quindi puoi distribuire solo il codice dell'applicazione e creare su istanza per evitare di raggruppare e caricare tutte le tue dipendenze.

Puoi utilizzare le proprietà dell'ambiente Elastic Beanstalk per configurare l'SDK X-Ray. Il metodo utilizzato da Elastic Beanstalk per passare le proprietà dell'ambiente all'applicazione varia in base alla piattaforma. Utilizza le variabili di ambiente o le proprietà di sistema di X-Ray SDK a seconda della piattaforma in uso.

- Piattaforma Node.js: utilizza le variabili di ambiente
- Piattaforma Java SE: utilizza variabili di ambiente
- Piattaforma Tomcat: utilizza le proprietà del sistema

Per ulteriori informazioni, consulta Configuring AWS X-Ray Debugging nella Developer Guide. AWS Elastic Beanstalk

Elastic Load Balancing e AWS X-Ray

I sistemi di bilanciamento del carico delle applicazioni Elastic Load Balancing aggiungono un ID di traccia alle richieste HTTP in entrata in un'intestazione denominata. `X-Amzn-Trace-Id`

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Formato ID di traccia X-Ray

Una radiografia `trace_id` è composta da tre numeri separati da trattini. Ad esempio, `1-58406520-a006649127e371903a2de979`. Questo include:

- Il numero di versione, che è. `1`
- L'ora della richiesta originale in Unix epoch time utilizzando 8 cifre esadecimali.

  Ad esempio, il fuso orario PST delle 10:00 del 1° dicembre 2016 è espresso in secondi o in cifre esadecimali. `1480615200` `58406520`
- Un identificatore a 96 bit univoco a livello globale per la traccia in 24 cifre esadecimali.

I sistemi di bilanciamento del carico non inviano dati a X-Ray e non vengono visualizzati come nodo sulla mappa dei servizi.

Per ulteriori informazioni, consulta Request Tracing for Your Application Load Balancer nella Elastic Load Balancing Developer Guide.

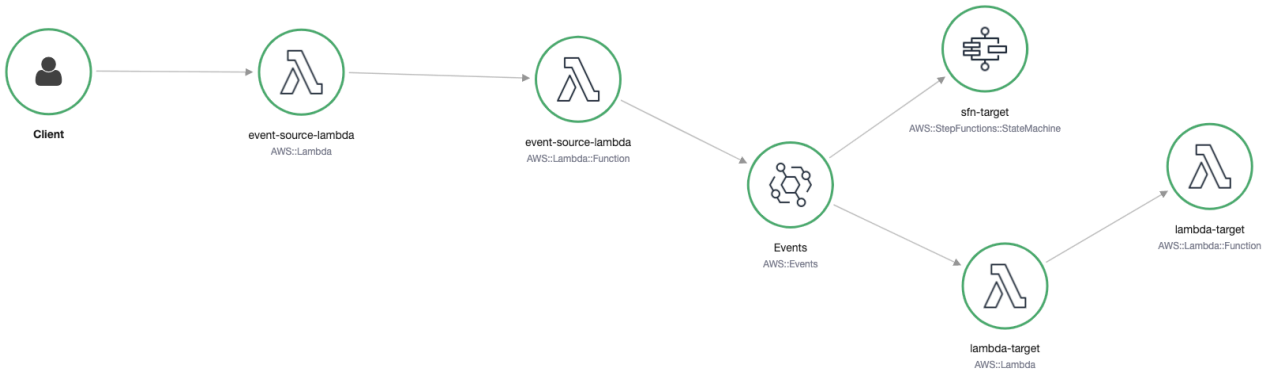
Amazon EventBridge e AWS X-Ray

AWS X-Ray si integra con Amazon EventBridge per tracciare gli eventi trasmessi EventBridge. Se un servizio strumentato con l' X-Ray SDK invia eventi a EventBridge, il contesto di traccia viene propagato alle destinazioni degli eventi a valle all'interno dell'intestazione di tracciamento. ??? L' X-Ray SDK preleva automaticamente l'intestazione di tracciamento e la applica a qualsiasi strumentazione successiva. Questa continuità consente agli utenti di tracciare, analizzare ed eseguire il debug di tutti i servizi a valle e fornisce una visione più completa del loro sistema.

Per ulteriori informazioni, vedere EventBridge X-Ray Integrazione nella Guida per l'EventBridge utente.

Visualizzazione dell'origine e delle destinazioni sulla mappa del X-Ray servizio

La mappa di X-Ray traccia mostra un nodo di EventBridge evento che collega i servizi di origine e di destinazione, come nell'esempio seguente:



Propaga il contesto di traccia alle destinazioni degli eventi

L' X-Ray SDK consente all'origine dell' EventBridge evento di propagare il contesto di traccia alle destinazioni degli eventi a valle. I seguenti esempi specifici della lingua dimostrano la chiamata EventBridge da una funzione Lambda su cui è abilitato il tracing attivo: https://docs.aws.amazon.com/lambda/latest/dg/services-xray.html#services-xray-api

Java

Aggiungi le dipendenze necessarie per: X-Ray

- AWS X-Ray SDK per Java
- AWS X-Ray Recorder SDK per Java

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.services.eventbridge.AmazonEventBridge;
import com.amazonaws.services.eventbridge.AmazonEventBridgeClientBuilder;
import com.amazonaws.services.eventbridge.model.PutEventsRequest;
import com.amazonaws.services.eventbridge.model.PutEventsRequestEntry;
import com.amazonaws.services.eventbridge.model.PutEventsResult;
import com.amazonaws.services.eventbridge.model.PutEventsResultEntry;
import com.amazonaws.xray.handlers.TracingHandler;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import java.lang.StringBuilder;
import java.util.Map;
import java.util.List;
import java.util.Date;
import java.util.Collections;

/*
   Add the necessary dependencies for XRay:
   https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-xray
   https://mvnrepository.com/artifact/com.amazonaws/aws-xray-recorder-sdk-aws-sdk
*/
public class Handler implements RequestHandler<SQSEvent, String>{
  private static final Logger logger = LoggerFactory.getLogger(Handler.class);

  /*
    build EventBridge client
  */
  private static final AmazonEventBridge eventsClient =
 AmazonEventBridgeClientBuilder
          .standard()
          // instrument the EventBridge client with the XRay Tracing Handler.
          // the AWSXRay globalRecorder will retrieve the tracing-context
          // from the lambda function and inject it into the HTTP header.
          // be sure to enable 'active tracing' on the lambda function.
          .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
          .build();

  @Override
  public String handleRequest(SQSEvent event, Context context)
  {
    PutEventsRequestEntry putEventsRequestEntry0 = new PutEventsRequestEntry();
    putEventsRequestEntry0.setTime(new Date());
    putEventsRequestEntry0.setSource("my-lambda-function");
    putEventsRequestEntry0.setDetailType("my-lambda-event");
    putEventsRequestEntry0.setDetail("{\"lambda-source\":\"sqs\"}");
    PutEventsRequest putEventsRequest = new PutEventsRequest();
    putEventsRequest.setEntries(Collections.singletonList(putEventsRequestEntry0));
    // send the event(s) to EventBridge
    PutEventsResult putEventsResult = eventsClient.putEvents(putEventsRequest);
    try {
      logger.info("Put Events Result: {}", putEventsResult);
    } catch(Exception e) {
      e.getStackTrace();
```

```
    }
    return "success";
  }
}
```

Python

Aggiungi la seguente dipendenza al tuo file 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#

Aggiungi i seguenti X-Ray pacchetti alle tue dipendenze 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 e AWS X-Ray

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi usarla per tracciare le tue funzioni. AWS X-Ray AWS Lambda Lambda esegue il demone X-Ray e registra un segmento con dettagli sull'invocazione e l'esecuzione della funzione. Per ulteriore strumentazione, puoi abbinare l'X-Ray SDK alla tua funzione per registrare le chiamate in uscita e aggiungere annotazioni e metadati.

Se la funzione Lambda viene chiamata da un altro servizio strumentato, Lambda traccia le richieste che sono già state campionate senza alcuna configurazione aggiuntiva. Il servizio upstream può essere un'applicazione Web strumentata o un'altra funzione Lambda. Il tuo servizio può richiamare la

funzione direttamente con un client AWS SDK strumentato o chiamando un'API API Gateway con un client HTTP strumentato.

AWS X-Ray supporta il tracciamento di applicazioni basate sugli eventi utilizzando Amazon AWS Lambda SQS. Usa la CloudWatch console per visualizzare una vista connessa di ogni richiesta mentre viene messa in coda con Amazon SQS ed elaborata da una funzione Lambda downstream. Le tracce dei produttori di messaggi upstream vengono automaticamente collegate alle tracce dei nodi consumer Lambda a valle, creando una end-to-end visualizzazione dell'applicazione. Per ulteriori informazioni, consulta la sezione di tracciamento delle applicazioni basate sugli eventi.

> **Note**
>
> Se hai le tracce abilitate per una funzione Lambda a valle, devi avere le tracce abilitate anche per la funzione Lambda principale che chiama la funzione downstream affinché la funzione a valle generi tracce.

Se la funzione Lambda viene eseguita in base a una pianificazione o viene richiamata da un servizio non dotato di strumentazione, è possibile configurare Lambda per campionare e registrare le chiamate con tracciamento attivo.

Per configurare l'integrazione X-Ray su una funzione AWS Lambda

1. Apri la AWS Lambda console.
2. Seleziona Funzioni dalla barra di navigazione a sinistra.
3. Scegliere la funzione.
4. Nella scheda Configurazione, scorri verso il basso fino alla scheda Strumenti di monitoraggio aggiuntivi. Puoi trovare questa scheda anche selezionando Strumenti di monitoraggio e operazioni nel riquadro di navigazione a sinistra.
5. Seleziona Edit (Modifica).
6. In AWS X-ray, abilitare Active tracing (Tracciamento attivo).

Nei runtime con un SDK X-Ray corrispondente, Lambda esegue anche il demone X-Ray.

X-Ray su SDKs Lambda

- X-Ray SDK for Go — Go 1.7 e runtime più recenti

- X-Ray SDK per Java — Java 8 runtime
- X-Ray SDK per Node.js — Node.js 4.3 e versioni successive
- X-Ray SDK per Python — Python 2.7, Python 3.6 e runtime più recenti
- X-Ray SDK for .NET: .NET Core 2.0 e runtime più recenti

Per utilizzare l'SDK X-Ray su Lambda, aggiungilo al codice funzione ogni volta che crei una nuova versione. Puoi strumentare le tue funzioni Lambda con gli stessi metodi che usi per strumentare le applicazioni in esecuzione su altri servizi. La differenza principale è che non si utilizza l'SDK per analizzare le richieste in entrata, assumere le decisioni di campionamento e creare i segmenti.

L'altra differenza tra la strumentazione delle funzioni Lambda e le applicazioni Web è che il segmento che Lambda crea e invia a X-Ray non può essere modificato dal codice della funzione. Puoi creare sottosegmenti e memorizzare in essi annotazioni e metadati, ma non puoi aggiungere annotazioni e metadati al segmento padre.

Per ulteriori informazioni, consulta Using AWS X-Ray nella AWS Lambda Developer Guide.

AWS Step Functions and AWS X-Ray

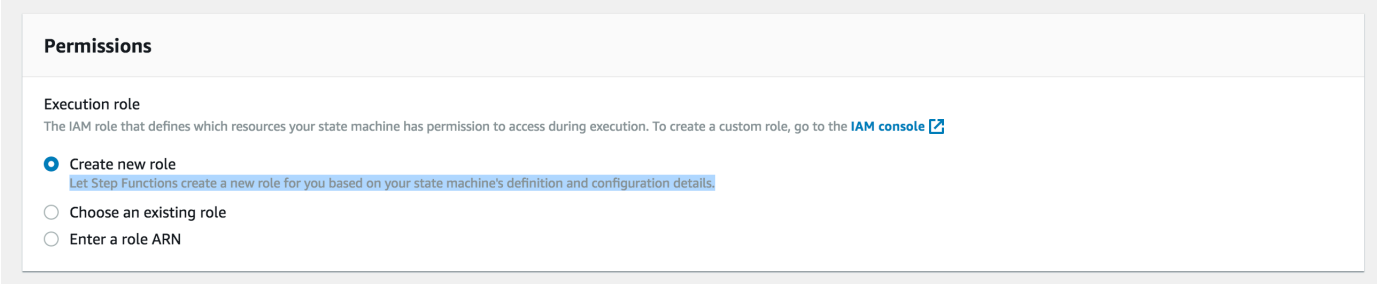
AWS X-Ray si integra con AWS Step Functions per tracciare e analizzare le richieste di Step Functions. Puoi visualizzare i componenti della tua macchina a stati, identificare i colli di bottiglia nelle prestazioni e risolvere le richieste che hanno provocato un errore. Per ulteriori informazioni, consulta la sezione Step Functions nella Guida per AWS X-Ray gli sviluppatori. AWS Step Functions

Per abilitare la X-Ray traccia durante la creazione di una nuova macchina a stati

1. Aprire la console Step Functions all'indirizzo. https://console.aws.amazon.com/states/
2. Scegli Crea una macchina a stati.
3. Nella pagina Definisci macchina a stati, scegli Autore con frammenti di codice o Inizia con un modello. Se scegli di eseguire un progetto di esempio, non puoi abilitare la X-Ray traccia durante la creazione. Attivate invece X-Ray la traccia dopo aver creato la vostra macchina a stati.
4. Scegli Avanti.
5. Nella pagina Specifica i dettagli, configura la tua macchina a stati.
6. Scegli Abilita X-Ray tracciamento.

Per abilitare la X-Ray traccia in una macchina a stati esistente

1. Nella console Step Functions, selezionate la macchina a stati per la quale desiderate abilitare la traccia.
2. Scegli Modifica.
3. Scegliete Abilita X-Ray tracciamento.
4. (Facoltativo) Auto-generate un nuovo ruolo per la macchina a stati che includa X-Ray le autorizzazioni scegliendo Crea nuovo ruolo dalla finestra Autorizzazioni.



5. Selezionare Salva.

> **Note**
>
> Quando crei una nuova macchina a stati, questa viene tracciata automaticamente se la richiesta viene campionata e la traccia è abilitata in un servizio upstream come Amazon API Gateway o. AWS Lambda Per qualsiasi macchina a stati esistente non configurata tramite la console, ad esempio tramite un CloudFormation modello, verifica di disporre di una policy IAM che conceda autorizzazioni sufficienti per abilitare le tracce. X-Ray

Strumentazione della tua applicazione per AWS X-Ray

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

La strumentazione dell'applicazione comporta l'invio di dati di traccia per le richieste in entrata e in uscita e altri eventi all'interno dell'applicazione, insieme ai metadati relativi a ciascuna richiesta. Esistono diverse opzioni di strumentazione tra cui scegliere o combinare, in base alle proprie esigenze particolari:

- Strumentazione automatica: strumentazione della tua applicazione senza modifiche al codice, in genere tramite modifiche alla configurazione, aggiunta di un agente di strumentazione automatica o altri meccanismi.
- Strumentazione della libreria: apporta modifiche minime al codice dell'applicazione per aggiungere strumentazione predefinita destinata a librerie o framework specifici, come SDK, client Apache HTTP o client SQL. AWS
- Strumentazione manuale: aggiungi il codice di strumentazione all'applicazione in ogni posizione in cui desideri inviare le informazioni di traccia.

Esistono diversi SDKs agenti e strumenti che possono essere utilizzati per strumentare l'applicazione per il tracciamento a raggi X..

Argomenti

- Strumentate la vostra applicazione con Distro per AWS OpenTelemetry
- Strumentate la vostra applicazione con AWS X-Ray SDKs
- Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs

Strumentate la vostra applicazione con Distro per AWS OpenTelemetry

The AWS Distro for OpenTelemetry (ADOT) è una AWS distribuzione basata sul progetto Cloud Native Computing Foundation (CNCF). OpenTelemetry OpenTelemetry fornisce un unico set di librerie APIs, agenti e librerie open source per raccogliere tracce e metriche distribuite. Questo toolkit è una distribuzione di OpenTelemetry componenti upstream SDKs, inclusi agenti di strumentazione automatica e collettori testati, ottimizzati, protetti e supportati da. AWS

Con ADOT, gli ingegneri possono strumentare le loro applicazioni una sola volta e inviare metriche e tracce correlate a più soluzioni di AWS monitoraggio tra cui Amazon CloudWatch AWS X-Ray e Amazon Service. OpenSearch

L'utilizzo di X-Ray con ADOT richiede due componenti: un OpenTelemetry SDK abilitato per l'uso con X-Ray e AWS Distro for OpenTelemetry Collector abilitato per l'uso con X-Ray. Per ulteriori informazioni sull'utilizzo di Distro for OpenTelemetry with AWS X-Ray e altro, AWS consulta Distro for Documentation. Servizi AWSAWS OpenTelemetry

Per ulteriori informazioni sul supporto e l'utilizzo della lingua, vedere AWS Observability on. GitHub

> **Note**
>
> Ora puoi utilizzare l' CloudWatch agente per raccogliere parametri, log e tracce dalle istanze Amazon EC2 e dai server locali. CloudWatch la versione dell'agente 1.300025.0 e successive può raccogliere tracce dal nostro client OpenTelemetryX-Ray SDKs e inviarle a X-Ray. L'utilizzo dell' CloudWatch agente al posto del demone AWS Distro for OpenTelemetry (ADOT) Collector o X-Ray per raccogliere le tracce può aiutare a ridurre il numero di agenti da gestire. Per ulteriori informazioni, consultate l'argomento relativo all'CloudWatch agente nella Guida per l'utente CloudWatch .

ADOT include quanto segue:

- AWS Distro for Go OpenTelemetry
- AWS Distro per Java OpenTelemetry
- AWS Distro per OpenTelemetry JavaScript
- AWS Distro per Python OpenTelemetry

- AWS Distro per.NET OpenTelemetry

ADOT attualmente include il supporto per la strumentazione automatica per Java e Python. Inoltre, ADOT consente la strumentazione automatica delle funzioni AWS Lambda e delle relative richieste downstream utilizzando i runtime Java, Node.js e Python, tramite ADOT Managed Lambda Layers.

ADOT SDKs for Java and Go supporta le regole di campionamento centralizzato X-Ray. Se hai bisogno di supporto per le regole di campionamento a raggi X in altre lingue, prendi in considerazione l'utilizzo di un SDK. AWS X-Ray

> **Note**
>
> Ora puoi inviare una traccia W3C a IDs X-Ray. Per impostazione predefinita, le tracce create con OpenTelemetry hanno un formato ID di traccia basato sulla specifica W3C Trace Context. Si tratta di un formato diverso dal formato di traccia IDs creato utilizzando un SDK X-Ray o dai AWS servizi integrati con X-Ray. Per garantire che le tracce IDs in formato W3C siano accettate da X-Ray, è necessario utilizzare la versione AWS X-Ray Exporter 0.86.0 o successiva, inclusa in ADOT Collector versione 0.34.0 e successive. Le versioni precedenti dell'esportatore convalidano i timestamp degli ID di traccia, il che potrebbe causare il rifiuto della traccia W3C. IDs

Strumentate la vostra applicazione con AWS X-Ray SDKs

AWS X-Ray include un set di linguaggi specifici SDKs per strumentare l'applicazione per inviare tracce a X-Ray. Ogni X-Ray SDK offre quanto segue:

- Moduli di intercettazione da aggiungere al tuo codice per il tracciamento delle richieste HTTP in ingresso
- Da gestori client a client AWS SDK di Instrument che l'applicazione utilizza per chiamare altri Servizi AWS
- Un client HTTP per effettuare chiamate strumentali ad altri servizi Web HTTP interni ed esterni

X-Ray supporta SDKs anche le chiamate strumentali ai database SQL, la strumentazione client AWS SDK automatica e altre funzionalità. Invece di inviare i dati di traccia direttamente a X-Ray, l'SDK invia i documenti del segmento JSON a un processo demone che ascolta il traffico UDP. Il demone X-Ray memorizza i segmenti in una coda e li carica su X-Ray in batch.

Sono SDKs disponibili le seguenti lingue specifiche:

- AWS X-Ray SDK for Go
- AWS X-Ray SDK per Java
- AWS X-Ray SDK per Node.js
- AWS X-Ray SDK per Python
- AWS X-Ray SDK per.NET
- AWS X-Ray SDK per Ruby

X-Ray attualmente include il supporto per la strumentazione automatica per Java.

Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs

Gli X-Ray SDKs inclusi fanno parte di una soluzione di strumentazione strettamente integrata offerta da. AWS AWS Distro for OpenTelemetry fa parte di una soluzione di settore più ampia in cui X-Ray è solo una delle tante soluzioni di tracciamento. È possibile implementare il end-to-end tracciamento in X-Ray utilizzando entrambi gli approcci, ma è importante comprendere le differenze per determinare l'approccio più utile per l'utente.

Ti consigliamo di strumentare la tua applicazione con AWS Distro OpenTelemetry se hai bisogno di quanto segue:

- La possibilità di inviare tracce a più backend di tracciamento diversi senza dover ristrumentare il codice
- Support per un gran numero di strumentazioni di libreria per ogni lingua, gestite dalla comunità OpenTelemetry
- Layer Lambda completamente gestiti che racchiudono tutto il necessario per raccogliere dati di telemetria, senza richiedere modifiche al codice quando si utilizza Java, Python o Node.js

> **Note**
>
> AWS Distro for OpenTelemetry offre un'esperienza introduttiva più semplice per la strumentazione delle funzioni Lambda. Tuttavia, a causa della flessibilità OpenTelemetry offerta, la funzione Lambda richiederà memoria aggiuntiva e le chiamate potrebbero subire un aumento della latenza di avvio a freddo, il che può comportare costi aggiuntivi. Se stai ottimizzando per una bassa latenza e non hai bisogno delle funzionalità avanzate

come destinazioni OpenTelemetry di backend configurabili dinamicamente, potresti voler utilizzare l'SDK AWS X-Ray per strumentare la tua applicazione.

Ti consigliamo di scegliere un SDK X-Ray per la strumentazione della tua applicazione se hai bisogno di quanto segue:

- Una soluzione monovendor strettamente integrata
- Integrazione con le regole di campionamento centralizzate X-Ray, inclusa la possibilità di configurare le regole di campionamento dalla console X-Ray e di utilizzarle automaticamente su più host, quando si utilizza Node.js, Python, Ruby o.NET

Transaction Search

Transaction Search è un'esperienza di analisi interattiva che puoi utilizzare per ottenere una visibilità completa della portata delle transazioni delle tue applicazioni. Gli intervalli sono le unità operative fondamentali in una traccia distribuita e rappresentano azioni o attività specifiche in un'applicazione o in un sistema. Ogni intervallo registra i dettagli su un particolare segmento della transazione. Questi dettagli includono l'ora di inizio e di fine, la durata e i metadati associati, che possono includere attributi aziendali come gli ID dei clienti e gli ID degli ordini. Gli intervalli sono organizzati in una gerarchia padre-figlio. Questa gerarchia forma una traccia completa che mappa il flusso di una transazione tra diversi componenti o servizi.

Per ulteriori informazioni, consulta Transaction Search.

OpenTelemetry Endpoint del protocollo (OTLP)

OpenTelemetry è un framework di osservabilità open source che fornisce ai team IT protocolli e strumenti standardizzati per la raccolta e il routing dei dati di telemetria. Fornisce un formato unificato per l'instrumentazione, la generazione, la raccolta e l'esportazione di dati di telemetria delle applicazioni, come metriche, log e tracce, verso piattaforme di monitoraggio per l'analisi e l'acquisizione di approfondimenti. Utilizzandolo OpenTelemetry, i team possono evitare il vincolo del fornitore, garantendo flessibilità nelle loro soluzioni di osservabilità.

Puoi utilizzarlo per inviare tracce direttamente OpenTelemetry a un endpoint di OpenTelemetry protocollo (OTLP) e ottenere esperienze di monitoraggio delle prestazioni delle applicazioni pronte all'uso in Application Signals. CloudWatch

Per ulteriori informazioni, consulta OpenTelemetry.

Utilizzo di Go

Esistono due modi per strumentare l'applicazione Go per inviare tracce a X-Ray:

- AWS Distro for OpenTelemetry Go: una AWS distribuzione che fornisce un set di librerie open source per l'invio di metriche e tracce correlate a più soluzioni di AWS monitoraggio tra cui Amazon AWS X-Ray e Amazon OpenSearch Service CloudWatch, tramite AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK for Go: un set di librerie per la generazione e l'invio di tracce a X-Ray tramite il demone X-Ray.

Per ulteriori informazioni, consulta Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs.

AWS Distro for OpenTelemetry Go

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Con AWS Distro for OpenTelemetry Go, puoi strumentare le tue applicazioni una sola volta e inviare metriche e tracce correlate a più soluzioni di AWS monitoraggio tra cui Amazon CloudWatch AWS X-Ray e Amazon Service. OpenSearch L'utilizzo di X-Ray con AWS Distro for OpenTelemetry richiede due componenti: un OpenTelemetry SDK abilitato per l'uso con X-Ray e AWS Distro for OpenTelemetry Collector abilitato per l'uso con X-Ray.

Per iniziare, consulta la documentazione di Distro for Go.AWS OpenTelemetry

Per ulteriori informazioni sull'utilizzo di AWS Distro for OpenTelemetry AWS X-Ray e altro Servizi AWS, consulta AWS Distro for OpenTelemetry o Distro for DocumentationAWS . OpenTelemetry

Per ulteriori informazioni sul supporto e l'utilizzo della lingua, vedere AWS Observability on. GitHub

AWS X-Ray SDK for Go

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L'X-Ray SDK for Go è un insieme di librerie per applicazioni Go che forniscono classi e metodi per generare e inviare dati di traccia al demone X-Ray. I dati di traccia includono informazioni sulle richieste HTTP in entrata fornite dall'applicazione e sulle chiamate che l'applicazione effettua ai servizi downstream utilizzando l' AWS SDK, i client HTTP o un connettore di database SQL. Puoi anche possibile creare manualmente dei segmenti e aggiungere informazioni di debug in annotazioni e metadati.

Scarica l'SDK dal suo repository con: GitHub `go get`

```
$ go get -u github.com/aws/aws-xray-sdk-go/...
```

In caso di applicazioni web, per tracciare le richieste in entrata inizia utilizzando la funzione `xray.Handler`. Il gestore dei messaggi crea un segmento per ogni richiesta tracciata e completa il segmento quando viene inviata la risposta. Fino a che il segmento è aperto puoi usare i metodi del client dell'SDK per aggiungere informazioni al segmento e creare sottosegmenti per tracciare le chiamate a valle. L'SDK, inoltre, registra automaticamente le eccezioni sollevate dall'applicazione per il tempo durante il quale il segmento è aperto.

Per le funzioni Lambda richiamate da un'applicazione o un servizio strumentato, Lambda legge l'intestazione di tracciamento e traccia automaticamente le richieste campionate. Per altre funzioni, puoi configurare Lambda per campionare e tracciare le richieste in arrivo. In entrambi i casi, Lambda crea il segmento e lo fornisce all'X-Ray SDK.

> **Note**
>
> Su Lambda, l'SDK X-Ray è opzionale. Se non lo usi nella tua funzione, la mappa dei servizi includerà comunque un nodo per il servizio Lambda e uno per ogni funzione Lambda. Aggiungendo l'SDK, puoi utilizzare il codice della funzione per aggiungere sottosegmenti al segmento di funzione registrato da Lambda. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Quindi, includi il tuo client all'interno di una chiamata alla funzione `AWS`. Questa fase assicura che X-Ray analizzi le chiamate verso qualsiasi metodo client. Puoi anche analizzare le chiamate verso database SQL.

Dopo aver iniziato a utilizzare l'SDK, personalizzane il comportamento configurando il registratore e il middleware. Puoi aggiungere dei plugin per memorizzare i dati sulle risorse di elaborazione sulle quali è eseguita la tua applicazione, personalizzare il comportamento di campionamento definendo regole di campionatura e impostare il livello di log per visualizzare più o meno informazioni generate dall'SDK nei log dell'applicazione.

Registra ulteriori informazioni sulle richieste e sull'attività svolta dalla tua applicazione in annotazioni e metadati. Le annotazioni sono semplici coppie chiave-valore indicizzati per l'uso con espressioni filtro, in modo da poter cercare tracce che contengono dati specifici. Le immissioni di metadati sono meno restrittive e possono registrare interi oggetti e array, ovvero tutto ciò che può essere serializzato in JSON.

> **Annotazioni e metadata**
>
> Le annotazioni e i metadati sono testo arbitrario che aggiungi ai segmenti con X-Ray SDK. Le annotazioni vengono indicizzate per essere utilizzate con le espressioni di filtro. I metadati non sono indicizzati, ma possono essere visualizzati nel segmento non elaborato con la console X-Ray o l'API. Chiunque conceda l'accesso in lettura a X-Ray può visualizzare questi dati.

Quando disponi di una notevole quantità di client analizzati nel tuo codice, un singolo segmento per la richiesta può contenere un numero elevato di sottosegmenti, uno per ciascuna delle chiamate effettuate con un client analizzato. Puoi organizzare e raggruppare i sottosegmenti inglobando le chiamate del client sottosegmenti personalizzati. Puoi creare un sottosegmento personalizzato

per un'intera funzione o qualsiasi porzione di codice, e memorizzare metadati e annotazioni sul sottosegmento invece di scrivere tutto all'interno del segmento padre.

Requisiti

L'X-Ray SDK for Go richiede Go 1.9 o versione successiva.

L'SDK dipende dalle seguenti librerie per la compilazione e il runtime:

- AWS SDK for Go versione 1.10.0 o successiva

Queste dipendenze sono dichiarate nel file `README.md` dell'SDK.

Documentazione di riferimento

Una volta scaricato l'SDK, crea e conserva la documentazione in locale per visualizzarla in un browser Web.

Per visualizzare la documentazione di riferimento

1. Spostarsi nella cartella `$GOPATH/src/github.com/aws/aws-xray-sdk-go` (Linux o Mac) o nella cartella `%GOPATH%\src\github.com\aws\aws-xray-sdk-go` (Windows)
2. Esegui il comando `godoc`.

   ```
   $ godoc -http=:6060
   ```

3. Aprire il browser all'indirizzo `http://localhost:6060/pkg/github.com/aws/aws-xray-sdk-go/`.

Configurazione dell' X-Ray SDK for Go

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS verranno limitate le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support
> Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione

a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

Puoi specificare la configurazione per X-Ray SDK for Go tramite variabili di ambiente, `Configure` chiamando con `Config` un oggetto o assumendo valori predefiniti. Le variabili di ambiente prevalgono sui valori di `Config`, che hanno la priorità su qualsiasi valore predefinito.

Sections

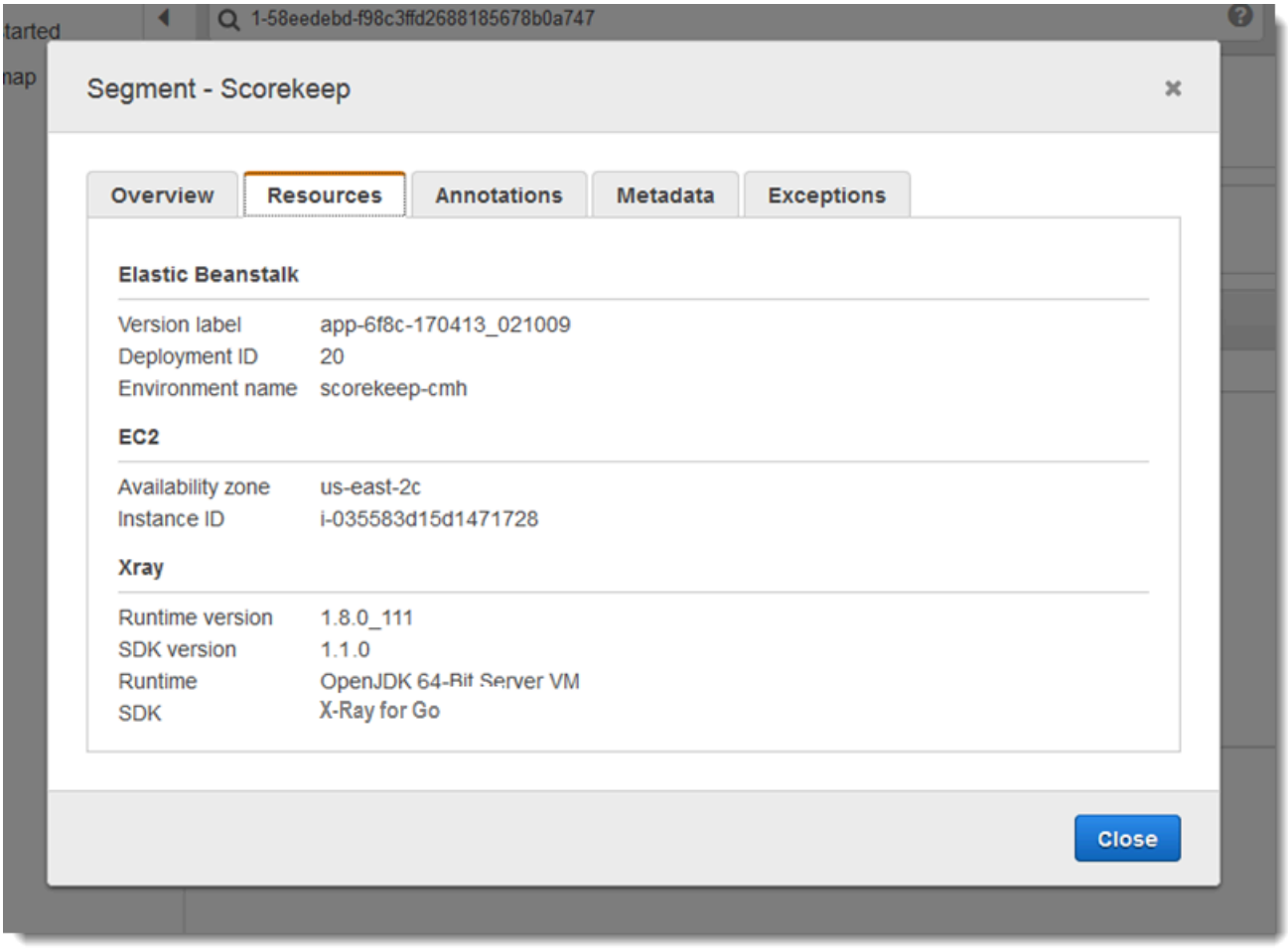
- Plugin di servizio
- Regole di campionamento
- Registrazione dei log
- Variabili di ambiente
- Utilizzo della configurazione

Plugin di servizio

`plugins`Utilizzatelo per registrare informazioni sul servizio che ospita l'applicazione.

Plugin

- Amazon EC2: `EC2Plugin` aggiunge l'ID dell'istanza, la zona di disponibilità e il gruppo di CloudWatch log.
- Elastic `ElasticBeanstalkPlugin` Beanstalk: aggiunge il nome dell'ambiente, l'etichetta della versione e l'ID di distribuzione.
- Amazon ECS: `ECSPlugin` aggiunge l'ID del contenitore.



Per usare un plugin, importare uno dei seguenti pacchetti.

```
"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"
```

Ogni plugin ha una chiamata di funzione `Init()` esplicita che carica il plugin.

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",
  })
}
```

L'SDK utilizza anche le impostazioni del plug-in per impostare il campo sul segmento. `origin` Indica il tipo di AWS risorsa che esegue l'applicazione. Quando utilizzi più plugin, l'SDK utilizza il seguente ordine di risoluzione per determinare l'origine: ElasticBeanstalk > EKS > ECS > EC2.

Regole di campionamento

L'SDK utilizza le regole di campionamento definite nella X-Ray console per determinare quali richieste registrare. La regola predefinita tiene traccia della prima richiesta ogni secondo e del 5% di tutte le richieste aggiuntive su tutti i servizi a cui vengono inviate le tracce. X-Ray Crea regole aggiuntive nella X-Ray console per personalizzare la quantità di dati registrati per ciascuna delle tue applicazioni.

L'SDK applica le regole personalizzate nell'ordine in cui sono definite. Se una richiesta corrisponde a più regole personalizzate, l'SDK applica solo la prima regola.

> **Note**
>
> Se l'SDK non riesce X-Ray a ottenere le regole di campionamento, torna a una regola locale predefinita che prevede la prima richiesta ogni secondo e il 5% di eventuali richieste aggiuntive per host. Ciò può verificarsi se l'host non dispone dell'autorizzazione per chiamare le API di campionamento o non può connettersi al X-Ray demone, che funge da proxy TCP per le chiamate API effettuate dall'SDK.

Puoi anche configurare l'SDK per caricare le regole di campionamento da un documento JSON. L'SDK può utilizzare le regole locali come backup per i casi in cui il X-Ray campionamento non è disponibile o utilizzare esclusivamente regole locali.

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
  }
}
```

Questo esempio definisce una regola personalizzata e una regola predefinita. La regola personalizzata applica una frequenza di campionamento del cinque percento senza alcun numero minimo di richieste da tracciare per i percorsi. `/api/move/` La regola predefinita tiene traccia della prima richiesta ogni secondo e del 10% delle richieste aggiuntive.

Lo svantaggio della definizione locale delle regole è che l'obiettivo fisso viene applicato da ciascuna istanza del registratore in modo indipendente, anziché essere gestito dal X-Ray servizio. Man mano che si implementano più host, la tariffa fissa si moltiplica, rendendo più difficile il controllo della quantità di dati registrati.

Sì AWS Lambda, non è possibile modificare la frequenza di campionamento. Se la funzione viene chiamata da un servizio strumentato, le chiamate che hanno generato richieste campionate da quel servizio verranno registrate da Lambda. Se il tracciamento attivo è abilitato e non è presente alcuna intestazione di tracciamento, Lambda prende la decisione di campionamento.

Per fornire le regole di backup, scegli il file JSON per il campionamento locale utilizzando `NewCentralizedStrategyWithFilePath`.

Example main.go — Regola di campionamento locale

```
s, _ := sampling.NewCentralizedStrategyWithFilePath("sampling.json") // path to local
 sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Per usare solo regole locali, seleziona il JSON per il campionamento locale utilizzando `NewLocalizedStrategyFromFilePath`.

Example main.go — Disabilita il campionamento

```
s, _ := sampling.NewLocalizedStrategyFromFilePath("sampling.json") // path to local
 sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Registrazione dei log

> **Note**
>
> I campi `xray.Config{}` `LogLevel` e `LogFormat` sono considerati obsoleti a partire dalla versione 1.0.0-rc.10.

X-Ray utilizza la seguente interfaccia per la registrazione. Il logger predefinito scrive su `stdout` a livello `LogLevelInfo` e superiori.

```
type Logger interface {
 Log(level LogLevel, msg fmt.Stringer)
}

const (
 LogLevelDebug LogLevel = iota + 1
 LogLevelInfo
 LogLevelWarn
 LogLevelError
)
```

Example **`scrivi su IO.Writer`**

```
xray.SetLogger(xraylog.NewDefaultLogger(os.Stderr, xraylog.LogLevelError))
```

Variabili di ambiente

Puoi utilizzare le variabili di ambiente per configurare l' X-Ray SDK for Go. L'SDK supporta le seguenti variabili.

- `AWS_XRAY_CONTEXT_MISSING`— Imposta `RUNTIME_ERROR` per generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.

  Valori validi

  - `RUNTIME_ERROR`— Genera un'eccezione di runtime.
  - `LOG_ERROR`— Registra un errore e continua (impostazione predefinita).
  - `IGNORE_ERROR`— Ignora l'errore e continua.

  Gli errori relativi a segmenti o sottosegmenti mancanti possono verificarsi quando si tenta di utilizzare un client con strumenti nel codice di avvio che viene eseguito quando non è aperta alcuna richiesta o nel codice che genera un nuovo thread.
- `AWS_XRAY_TRACING_NAME`— Imposta il nome del servizio che l'SDK utilizza per i segmenti.
- `AWS_XRAY_DAEMON_ADDRESS`— Imposta l'host e la porta del X-Ray daemon listener. Per impostazione predefinita, l'SDK invia i dati di traccia a. `127.0.0.1:2000` Utilizzate questa variabile se avete configurato il demone per l'ascolto su una porta diversa o se è in esecuzione su un host diverso.

Le variabili di ambiente sostituiscono i valori equivalenti impostati nel codice.

Utilizzo della configurazione

Puoi anche configurare l' X-Ray SDK for Go utilizzando `Configure` il metodo. `Configure`accetta un argomento, un `Config` oggetto, con i seguenti campi opzionali.

DaemonAddr

Questa stringa specifica l'host e la porta del X-Ray daemon listener. Se non viene specificato, X-Ray utilizza il valore della variabile di ambiente. `AWS_XRAY_DAEMON_ADDRESS` Se il valore non è impostato, utilizza "127.0.0.1:2000".

ServiceVersion

Questa stringa specificata la versione del servizio. Se non specificato, X-Ray utilizza la stringa vuota («»).

SamplingStrategy

Questo oggetto `SamplingStrategy` specifica quali delle chiamate della tua applicazione sono tracciate. Se non viene specificato, X-Ray utilizza a`LocalizedSamplingStrategy`, che utilizza la strategia definita in`xray/resources/DefaultSamplingRules.json`.

StreamingStrategy

Questo `StreamingStrategy` oggetto specifica se eseguire lo streaming di un segmento quando RequiresStreamingrestituisce true. Se non è specificato, X-Ray utilizza una `DefaultStreamingStrategy` che esegue lo streaming di un segmento campionato se il numero di sottosegmenti è superiore a 20.

ExceptionFormattingStrategy

Questo oggetto `ExceptionFormattingStrategy` specifica il modo in cui desideri gestire varie eccezioni. Se non specificato, X-Ray utilizza un tipo `DefaultExceptionFormattingStrategy` con un `XrayError` o`error`, il messaggio di errore e lo stack trace.

Strumentazione delle richieste HTTP in arrivo con X-Ray SDK for Go

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi utilizzare l'SDK X-Ray per tracciare le richieste HTTP in entrata che la tua applicazione serve su un'istanza EC2 in Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Utilizza `xray.Handler` per analizzare le richieste HTTP in entrata. L'X-Ray SDK for Go implementa l'`http.Handler`interfaccia di libreria Go standard nella `xray.Handler` classe per intercettare le richieste Web. La classe `xray.Handler` esegue il wrapping del gestore `http.Handler` fornito con `xray.Capture` utilizzando il contesto della richiesta, analizzando le intestazioni in ingresso, aggiungendo le intestazioni di risposta se necessario e impostando i campi HTTP specifici per il tracciamento.

Quando si utilizza questa classe per gestire le richieste e le risposte HTTP, X-Ray SDK for Go crea un segmento per ogni richiesta campionata. Questo segmento include durata, metodo e conclusione della richiesta HTTP. Analisi ulteriori creano sottosegmenti associati a questo segmento.

> **Note**
>
> Per quanto riguarda AWS Lambda le funzioni, Lambda crea un segmento per ogni richiesta campionata. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

L'esempio seguente intercetta le richieste sulla porta 8000 e restituisce «Hello!» come risposta. Crea il segmento `myApp` e le chiamate di analisi tramite qualsiasi applicazione.

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)
}
```

Ogni segmento ha un nome che identifica l'applicazione nella mappa dei servizi. Il segmento può essere denominato staticamente oppure è possibile configurare l'SDK per denominarlo dinamicamente in base all'intestazione dell'host nella richiesta in entrata. La denominazione dinamica consente di raggruppare le tracce in base al nome di dominio nella richiesta e di applicare un nome predefinito se il nome non corrisponde a uno schema previsto (ad esempio, se l'intestazione dell'host è falsificata).

> **Richieste inoltrate**
>
> Se un sistema di bilanciamento del carico o un altro intermediario inoltra una richiesta all'applicazione, X-Ray prende l'IP del client dall'`X-Forwarded-For`intestazione della richiesta anziché dall'IP di origine nel pacchetto IP. L'IP del client registrato per una richiesta inoltrata può essere falsificato, quindi non dovrebbe essere considerato attendibile.

Quando viene inoltrata una richiesta, l'SDK imposta un campo aggiuntivo nel segmento per indicarlo. Se il segmento contiene il campo `x_forwarded_for` impostato su`true`, l'IP del client è stato preso dall'`X-Forwarded-For`intestazione della richiesta HTTP.

Il gestore crea un segmento per ogni richiesta in entrata con un blocco `http` che contiene le informazioni riportate qui di seguito:

- Metodo HTTP: GET, POST, PUT, DELETE, ecc.
- Indirizzo client: l'indirizzo IP del client che ha inviato la richiesta.
- Codice di risposta: il codice di risposta HTTP per la richiesta completata.
- Tempistica: l'ora di inizio (quando è stata ricevuta la richiesta) e l'ora di fine (quando è stata inviata la risposta).
- Agente utente: il `user-agent` codice della richiesta.
- Lunghezza del contenuto: il `content-length` risultato della risposta.

Configurazione di una strategia di denominazione dei segmenti

AWS X-Ray utilizza un nome di servizio per identificare l'applicazione e distinguerla dalle altre applicazioni, database APIs, AWS risorse esterne e utilizzate dall'applicazione. Quando X-Ray SDK genera segmenti per le richieste in entrata, registra il nome del servizio dell'applicazione nel campo del nome del segmento.

L'X-Ray SDK può denominare i segmenti dopo il nome host nell'intestazione della richiesta HTTP. Tuttavia, questa intestazione può essere falsificata, il che potrebbe causare nodi imprevisti nella mappa dei servizi. Per evitare che l'SDK nomini i segmenti in modo errato a causa di richieste con intestazioni host contraffatte, è necessario specificare un nome predefinito per le richieste in entrata.

Se la tua applicazione soddisfa le richieste per più domini, puoi configurare l'SDK in modo che utilizzi una strategia di denominazione dinamica che rifletta questo aspetto nei nomi dei segmenti. Una strategia di denominazione dinamica consente all'SDK di utilizzare il nome host per le richieste che corrispondono a uno schema previsto e di applicare il nome predefinito alle richieste che non lo fanno.

Ad esempio, potresti avere una singola applicazione che serve le richieste a tre sottodomini:, e. `www.example.com` `api.example.com` `static.example.com` È possibile utilizzare una strategia di denominazione dinamica con lo schema `*.example.com` per identificare i segmenti per ogni sottodominio con un nome diverso, ottenendo tre nodi di servizio sulla mappa dei servizi. Se l'applicazione riceve richieste con un nome host che non corrisponde allo schema, sulla mappa dei servizi verrà visualizzato un quarto nodo con un nome di fallback specificato dall'utente.

Per utilizzare lo stesso nome per tutti i segmenti della richiesta, specifica il nome della tua applicazione quando crei il gestore, come illustrato nella sezione precedente.

> **Note**
>
> È possibile sovrascrivere il nome di servizio predefinito definito nel codice con la `AWS_XRAY_TRACING_NAME` variabile di ambiente.

Una strategia di denominazione dinamica definisce un modello al quale devono corrispondere i nomi degli host e un nome di default per l'utilizzo qualora il nome dell'host nella richiesta HTTP non corrisponda al modello. Per denominare segmenti in modo dinamico, utilizza `NewDynamicSegmentNamer` per configurare il nome di default e modello per la corrispondenza.

Example main.go

Se il nome host nella richiesta corrispondente al modello `*.example.com`, utilizza il nome dell'host. In caso contrario, utilizzare `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)
}
```

Tracciamento delle chiamate AWS SDK con X-Ray SDK for Go

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate per Servizi AWS archiviare dati, scrivere in una coda o inviare notifiche, X-Ray SDK for Go tiene traccia delle chiamate downstream in sottosegmenti. Le

risorse tracciate Servizi AWS e a cui accedi all'interno di tali servizi (ad esempio, un bucket Amazon S3 o una coda Amazon SQS) vengono visualizzate come nodi downstream sulla mappa di traccia nella console X-Ray.

Per tracciare i client AWS SDK, ingloba l'oggetto client nella chiamata `xray.AWS()` come mostrato nel seguente esempio.

Example main.go

```
var dynamo *dynamodb.DynamoDB
func main() {
  dynamo = dynamodb.New(session.Must(session.NewSession()))
  xray.AWS(dynamo.Client)
}
```

Quindi, quando utilizzi il client SDK AWS, utilizza la versione `withContext` del metodo chiamato e passa ad essa il `context` ricavato dall'oggetto `http.Request` passato al gestore.

Example main.go — chiamata SDK AWS

```
func listTablesWithContext(ctx context.Context) {
  output := dynamo.ListTablesWithContext(ctx, &dynamodb.ListTablesInput{})
  doSomething(output)
}
```

Per tutti i servizi, puoi vedere il nome dell'API richiamata nella console X-Ray. Per un sottoinsieme di servizi, l'SDK X-Ray aggiunge informazioni al segmento per fornire una maggiore granularità nella mappa dei servizi.

Ad esempio, quando si effettua una chiamata con un client DynamoDB con strumentazione, l'SDK aggiunge il nome della tabella al segmento per le chiamate destinate a una tabella. Nella console, ogni tabella appare come un nodo separato nella mappa dei servizi, con un nodo DynamoDB generico per le chiamate che non hanno come destinazione una tabella.

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
```

```
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Quando si accede alle risorse con nome, le chiamate ai seguenti servizi creano ulteriori nodi della mappa del servizio. Le chiamate che non sono hanno come obiettivo risorse specifiche creano un nodo generico per il servizio.

- Amazon DynamoDB: nome della tabella
- Amazon Simple Storage Service: nome del bucket e della chiave
- Amazon Simple Queue Service: nome della coda

Tracciamento delle chiamate ai servizi Web HTTP downstream con X-Ray SDK for Go

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate a microservizi o al protocollo HTTP pubblico APIs, è possibile utilizzare tali chiamate come sottosegmenti dell'*`xray.Client`*applicazione Go, come illustrato nell'esempio seguente, in cui http-client è un client HTTP.

Il client crea una copia superficiale del client HTTP fornito, l'impostazione predefinita è, con roundtripper racchiuso con. `http.DefaultClient` `xray.RoundTripper`

Example

<caption>main.go — Client HTTP</caption>

```
myClient := xray.Client(http-client)
```

<caption>main.go — Traccia la chiamata HTTP downstream con la libreria ctxhttp</caption>

L'esempio seguente riporta la chiamata HTTP in uscita con la libreria ctxhttp utilizzando. `xray.Client` `ctx`può essere passato dalla chiamata upstream. Ciò garantisce che venga utilizzato il contesto del segmento esistente. Ad esempio, X-Ray non consente la creazione di un nuovo segmento all'interno di una funzione Lambda, quindi è necessario utilizzare il contesto del segmento Lambda esistente.

```
resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

Tracciamento delle query SQL con X-Ray SDK for Go

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Per tracciare le chiamate SQL verso PostgreSQL o MySQL, sostituisci le chiamate a `sql.Open` con chiamate a `xray.SQLContext`, come nell'esempio seguente. Se URLs possibile, utilizzate al posto delle stringhe di configurazione.

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
}
```

Generazione di sottosegmenti personalizzati con X-Ray SDK for Go

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS verranno limitate le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support
> Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

I sottosegmenti estendono il segmento di una traccia con dettagli sul lavoro svolto per soddisfare una richiesta. Ogni volta che effettui una chiamata con un client dotato di strumenti, l' X-Ray SDK registra le informazioni generate in un sottosegmento. Puoi creare sottosegmenti aggiuntivi per raggruppare altri sottosegmenti, misurare le prestazioni di una sezione di codice o registrare annotazioni e metadati.

Utilizza il metodo `Capture` per creare un sottosegmento attinente a una funzione.

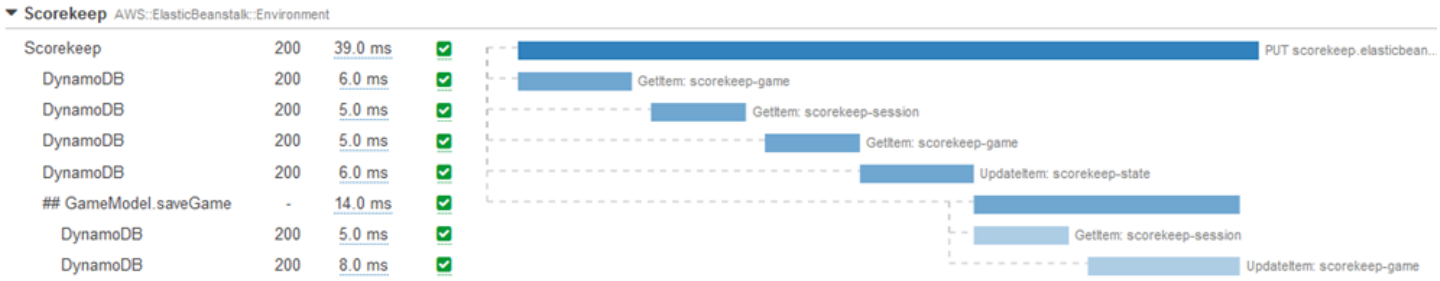
Example main.go — Sottosegmento personalizzato

```
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)
  })
```

La seguente schermata mostra un esempio di come il sottosegmento `saveGame` può essere visualizzato nei tracciamenti per l'applicazione `Scorekeep`.



Aggiungi annotazioni e metadati ai segmenti con X-Ray SDK for Go

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile registrare informazioni aggiuntive sulle richieste, sull'ambiente o sull'applicazione con annotazioni e metadati. È possibile aggiungere annotazioni e metadati ai segmenti creati da X-Ray SDK o ai sottosegmenti personalizzati creati dall'utente.

Le annotazioni sono coppie chiave-valore con stringhe, numeri o valori booleani. Le annotazioni sono indicizzate per essere utilizzate con le espressioni di filtro. Utilizzale per registrare i dati che desideri utilizzare per raggruppare le tracce nella console oppure per chiamare l'API `GetTraceSummaries`.

I metadati sono coppie chiave-valore che possono avere valori di qualsiasi tipo, inclusi oggetti ed elenchi, ma non sono indicizzati per essere utilizzati con le espressioni di filtro. Utilizzate i metadati per registrare dati aggiuntivi che desiderate archiviare nella traccia ma che non è necessario utilizzare con la ricerca.

Oltre ad annotazioni e metadati, sui segmenti puoi anche registrare le stringhe degli ID utente. IDs Gli utenti vengono registrati in un campo separato sui segmenti e indicizzati per essere utilizzati nella ricerca.

Sections

- Registrazione delle annotazioni con X-Ray SDK for Go

- Registrazione di metadati con X-Ray SDK for Go
- Utente che registra IDs con X-Ray SDK for Go

Registrazione delle annotazioni con X-Ray SDK for Go

Utilizza le annotazioni per memorizzare le informazioni sui segmenti che desideri siano indicizzate per la ricerca.

Requisiti per le annotazioni

- Chiavi: la chiave per un'annotazione a raggi X può contenere fino a 500 caratteri alfanumerici. Non è possibile utilizzare spazi o simboli diversi da un punto o un punto (.)
- Valori: il valore di un'annotazione X-Ray può contenere fino a 1.000 caratteri Unicode.
- Il numero di annotazioni: è possibile utilizzare fino a 50 annotazioni per traccia.

Per registrare le annotazioni, invoca `AddAnnotation` con una stringa che contiene i metadati che desideri associare al segmento.

```
xray.AddAnnotation(key string, value interface{})
```

L'SDK memorizza le annotazioni come coppie chiave-valore in un oggetto `annotations` all'interno del documento di segmento. Chiamando `AddAnnotation` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento viene sovrascritto.

Per trovare tracciamenti con annotazioni contenenti valori specifici, utilizza la parola chiave `annotation[`*`key`*`]` in un'espressione filtro.

Registrazione di metadati con X-Ray SDK for Go

Utilizza i metadati per memorizzare le informazioni su segmenti che non è necessario che siano indicizzate per la ricerca.

Per registrare i metadati, invoca `AddMetadata` con una stringa che contiene i metadati che desideri associare al segmento.

```
xray.AddMetadata(key string, value interface{})
```

Utente che registra IDs con X-Ray SDK for Go

Registra i segmenti dell'utente IDs su richiesta per identificare l'utente che ha inviato la richiesta.

Per registrare l'utente IDs

1. Ottenere un riferimento al segmento corrente da `AWSXRay`.

```
import (
  "context"
  "github.com/aws/aws-xray-sdk-go/xray"
)

mySegment := xray.GetSegment(context)
```

2. Chiamare `setUser` con una stringa che rappresenta l'ID dell'utente che ha inviato la richiesta.

```
mySegment.User = "U12345"
```

Per trovare tracciamenti associati ad un ID utente, utilizza la parola chiave `user` in un'espressione filtro.

Utilizzo di Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Esistono due modi per strumentare un'applicazione Java per inviare tracce a X-Ray:

- AWS Distro for OpenTelemetry Java: una AWS distribuzione che fornisce un set di librerie open source per l'invio di metriche e tracce correlate a più soluzioni di AWS monitoraggio, tra cui Amazon AWS X-Ray e Amazon OpenSearch Service CloudWatch, tramite AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK for Java: un set di librerie per la generazione e l'invio di tracce a X-Ray tramite il demone X-Ray.

Per ulteriori informazioni, consulta Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs.

AWS Distro per Java OpenTelemetry

Con AWS Distro for OpenTelemetry (ADOT) Java, puoi strumentare le tue applicazioni una sola volta e inviare metriche e tracce correlate a più AWS soluzioni di monitoraggio tra cui Amazon AWS X-Ray e Amazon CloudWatch Service. OpenSearch L'utilizzo di X-Ray con ADOT richiede due componenti: un OpenTelemetry SDK abilitato per l'uso con X-Ray e AWS Distro for OpenTelemetry Collector abilitato per l'uso con X-Ray. ADOT Java include il supporto per la strumentazione automatica, che consente all'applicazione di inviare tracce senza modifiche al codice.

Per iniziare, consultate la documentazione di AWS Distro for Java. OpenTelemetry

Per ulteriori informazioni sull'utilizzo di AWS Distro for OpenTelemetry AWS X-Ray e altro Servizi AWS, consulta AWS Distro for OpenTelemetry o Distro for DocumentationAWS . OpenTelemetry

Per ulteriori informazioni sul supporto e l'utilizzo della lingua, vedere AWS Observability on. GitHub

AWS X-Ray SDK per Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L'X-Ray SDK for Java è un insieme di librerie per applicazioni Web Java che forniscono classi e metodi per generare e inviare dati di traccia al demone X-Ray. I dati di traccia includono informazioni sulle richieste HTTP in entrata fornite dall'applicazione e sulle chiamate effettuate dall'applicazione ai servizi downstream utilizzando l' AWS SDK, i client HTTP o un connettore di database SQL. Puoi anche possibile creare manualmente dei segmenti e aggiungere informazioni di debug in annotazioni e metadati.

X-Ray SDK for Java è un progetto open source. Puoi seguire il progetto e inviare problemi e richieste pull su GitHub: github. com/aws/aws- xray-sdk-java

Inizia aggiungendo `AWSXRayServletFilter` come filtro del servlet per tracciare le richieste in entrata. Un filtro servlet crea un segmento. Fino a che il segmento è aperto, puoi usare i metodi del client dell'SDK per aggiungere informazioni al segmento e creare sottosegmenti per tracciare le chiamate a valle. L'SDK, inoltre, registra automaticamente le eccezioni sollevate dall'applicazione per il tempo durante il quale il segmento è aperto.

A partire dalla versione 1.3, è possibile analizzare l'applicazione utilizzando il sistema aspect-oriented programming (AOP) in Spring. Ciò significa che è possibile strumentare l'applicazione mentre è in esecuzione AWS, senza aggiungere alcun codice al runtime dell'applicazione.

Successivamente, utilizzate l'SDK X-Ray per Java per strumentare AWS SDK per Java i vostri client includendo il sottomodulo SDK Instrumentor nella configurazione di build. Ogni volta che effettui una chiamata a un downstream Servizio AWS o a una risorsa con un client strumentato, l'SDK registra le informazioni sulla chiamata in un sottosegmento. Servizi AWS e le risorse a cui accedi all'interno dei

servizi vengono visualizzate come nodi a valle sulla mappa di traccia per aiutarti a identificare errori e problemi di limitazione sulle singole connessioni.

Se non vuoi strumentalizzare tutte le chiamate downstream Servizi AWS, puoi tralasciare il sottomodulo Instrumentor e scegliere quali client strumentare. Strumenta i singoli client aggiungendo un client di servizio `TracingHandler` SDK. AWS

Altri sottomoduli X-Ray SDK for Java forniscono la strumentazione per le chiamate downstream ai database HTTP Web e SQL. APIs È possibile utilizzare le versioni X-Ray SDK for Java di `HTTPClient` e `HTTPClientBuilder` nel sottomodulo HTTP Apache per strumentare i client HTTP Apache. Per analizzare le query SQL, aggiungi lo strumento di intercettazione dell'SDK all'origine dei dati.

Dopo aver iniziato a utilizzare l'SDK, personalizzane il comportamento configurando il registratore e il filtro servlet. Puoi aggiungere dei plugin per memorizzare i dati sulle risorse di elaborazione sulle quali è eseguita la tua applicazione, personalizzare il comportamento di campionamento definendo regole di campionatura e impostare il livello di log per visualizzare più o meno informazioni generate dall'SDK nei log dell'applicazione.

Registra ulteriori informazioni sulle richieste e sull'attività svolta dalla tua applicazione in annotazioni e metadati. Le annotazioni sono semplici coppie chiave-valore indicizzati per l'uso con espressioni filtro, in modo da poter cercare tracce che contengono dati specifici. Le immissioni di metadati sono meno restrittive e possono registrare interi oggetti e array, ovvero tutto ciò che può essere serializzato in JSON.

> **Annotazioni e metadata**
>
> Le annotazioni e i metadati sono testo arbitrario che aggiungi ai segmenti con X-Ray SDK. Le annotazioni vengono indicizzate per essere utilizzate con le espressioni di filtro. I metadati non sono indicizzati, ma possono essere visualizzati nel segmento non elaborato con la console X-Ray o l'API. Chiunque conceda l'accesso in lettura a X-Ray può visualizzare questi dati.

Quando disponi di una notevole quantità di client analizzati nel tuo codice, un singolo segmento per la richiesta può contenere molti sottosegmenti, uno per ciascuna delle chiamate effettuate con un client analizzato. Puoi organizzare e raggruppare i sottosegmenti inglobando le chiamate del client sottosegmenti personalizzati. Puoi creare un sottosegmento personalizzato per un'intera funzione

o qualsiasi porzione di codice, e memorizzare metadati e annotazioni sul sottosegmento invece di scrivere tutto all'interno del segmento padre.

Sottomoduli

Puoi scaricare l'X-Ray SDK per Java da Maven. L'X-Ray SDK for Java è suddiviso in sottomoduli per caso d'uso, con una distinta dei materiali per la gestione delle versioni:

- `aws-xray-recorder-sdk-core`(obbligatorio): funzionalità di base per la creazione e la trasmissione di segmenti. Include `AWSXRayServletFilter` per l'analisi delle richieste in entrata.
- `aws-xray-recorder-sdk-aws-sdk`— Instruments: chiamate da Servizi AWS effettuare con AWS SDK per Java i clienti aggiungendo un client di tracciamento come gestore delle richieste.
- `aws-xray-recorder-sdk-aws-sdk-v2`— Chiamate Instruments Servizi AWS effettuate con client AWS SDK per Java 2.2 e versioni successive aggiungendo un client di tracciamento come intercettore delle richieste.
- `aws-xray-recorder-sdk-aws-sdk-instrumentor`— Con`aws-xray-recorder-sdk-aws-sdk`, Instruments tutti i client vengono automaticamente. AWS SDK per Java
- `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor`— Con`aws-xray-recorder-sdk-aws-sdk-v2`, instruments tutti i client AWS SDK per Java 2.2 e versioni successive automaticamente.
- `aws-xray-recorder-sdk-apache-http`— Chiamate HTTP in uscita di Instruments effettuate con client HTTP Apache.
- `aws-xray-recorder-sdk-spring`— Fornisce intercettori per le applicazioni Spring AOP Framework.
- `aws-xray-recorder-sdk-sql-postgres`— Instruments: chiamate in uscita a un database PostgreSQL realizzate con JDBC.
- `aws-xray-recorder-sdk-sql-mysql`— Chiamate in uscita di Instruments a un database MySQL realizzate con JDBC.
- `aws-xray-recorder-sdk-bom`— Fornisce una distinta dei materiali che è possibile utilizzare per specificare la versione da utilizzare per tutti i sottomoduli.
- `aws-xray-recorder-sdk-metrics`— Pubblica CloudWatch metriche Amazon non campionate dai segmenti X-Ray raccolti.

Se usi Maven o Gradle per creare la tua applicazione, aggiungi l'X-Ray SDK for Java alla configurazione di build.

Per la documentazione di riferimento sulle classi e sui metodi dell'SDK, consulta AWS X-Ray SDK for Java API Reference.

Requisiti

L'X-Ray SDK per Java richiede Java 8 o versione successiva, Servlet API 3, l' AWS SDK e Jackson.

L'SDK dipende dalle seguenti librerie per la compilazione e il runtime:

- AWS SDK for Java versione 1.11.398 o successiva
- API Servlet 3.1.0

Queste dipendenze sono dichiarate nel file `pom.xml` dell'SDK e vengono automaticamente incluse se compili utilizzando Maven o Gradle.

Se si utilizza una libreria inclusa in X-Ray SDK for Java, è necessario utilizzare la versione inclusa. Ad esempio, se fai già affidamento su Jackson in fase di runtime e includi dei file JAR nella distribuzione per tale dipendenza, devi rimuovere i file JAR perché il JAR dell'SDK include le proprie versioni delle librerie Jackson.

Gestione delle dipendenze

L'X-Ray SDK per Java è disponibile presso Maven:

- Gruppo — `com.amazonaws`
- Artefatto — `aws-xray-recorder-sdk-bom`
- Version (Versione): `2.11.0`

Se utilizzi Maven per compilare l'applicazione, aggiungi l'SDK come dipendenza nel tuo file `pom.xml`.

Example pom.xml - dipendenze

```
<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>
```

Nel caso di Gradle, aggiungi l'SDK come dipendenza in fase di compilazione nel tuo file `build.gradle`.

Example build.gradle - dipendenze

```
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 utilizzi Elastic Beanstalk per distribuire la tua applicazione, puoi usare Maven o Gradle per creare su istanza ogni volta che esegui la distribuzione, invece di creare e caricare un archivio di grandi dimensioni che include tutte le tue dipendenze. Vedi l'applicazione di esempio per un esempio che utilizza Gradle.

AWS X-Ray agente di strumentazione automatica per Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L'agente di AWS X-Ray strumentazione automatica per Java è una soluzione di tracciamento che strumenta le applicazioni Web Java con uno sforzo di sviluppo minimo. L'agente consente il tracciamento per le applicazioni basate su servlet e tutte le richieste downstream dell'agente effettuate con framework e librerie supportati. Ciò include le richieste HTTP Apache downstream, le richieste AWS SDK e le query SQL eseguite utilizzando un driver JDBC. L'agente propaga il contesto X-Ray, inclusi tutti i segmenti e i sottosegmenti attivi, tra i thread. Tutte le configurazioni e la versatilità dell'SDK X-Ray sono ancora disponibili con l'agente Java. Sono state scelte impostazioni predefinite adeguate per garantire che l'agente funzioni con il minimo sforzo.

La soluzione X-Ray Agent è la soluzione ideale per server di applicazioni Web Java basati su servlet e richiesta-risposta. Se la tua applicazione utilizza un framework asincrono o non è ben modellata

come servizio di richiesta-risposta, potresti prendere in considerazione la strumentazione manuale con l'SDK.

L'agente X-Ray è costruito utilizzando il toolkit Distributed Systems Comprehension, o Di. SCo Di SCo è un framework open source per la creazione di agenti Java che possono essere utilizzati in sistemi distribuiti. Sebbene non sia necessario conoscere Di SCo per utilizzare l'agente X-Ray, puoi saperne di più sul progetto visitando la sua home page all'indirizzo. GitHub L'agente X-Ray è inoltre completamente open source. Per visualizzare il codice sorgente, contribuire o sollevare problemi sull'agente, visita il suo repository su. GitHub

Applicazione di esempio

L'applicazione eb-java-scorekeepdi esempio è adattata per essere strumentata con l'agente X-Ray. Questo ramo non contiene alcuna configurazione del filtro servlet o del registratore, poiché queste funzioni vengono eseguite dall'agente. Per eseguire l'applicazione localmente o utilizzando AWS le risorse, seguite i passaggi nel file readme dell'applicazione di esempio. Le istruzioni per utilizzare l'app di esempio per generare tracce X-Ray si trovano nel tutorial dell'app di esempio.

Nozioni di base

Per iniziare a usare l'agente Java per la strumentazione automatica X-Ray nella tua applicazione, segui questi passaggi.

1. Esegui il demone X-Ray nel tuo ambiente. Per ulteriori informazioni, consulta AWS X-Ray demone.
2. Scarica la distribuzione più recente dell'agente. Decomprimi l'archivio e annota la sua posizione nel file system. Il suo contenuto dovrebbe essere simile al seguente.

   ```
   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. Modifica gli argomenti JVM dell'applicazione includendo quanto segue, che abilita l'agente. Assicuratevi che l'`-javaagent`argomento sia posto prima dell'`-jar`argomento, se applicabile. Il processo di modifica degli argomenti JVM varia a seconda degli strumenti e dei framework utilizzati per avviare il server Java. Consultate la documentazione del framework del vostro server per indicazioni specifiche.

```
-javaagent:/<path-to-disco>/disco-java-agent.jar=pluginPath=/<path-to-disco>/disco-plugins
```

4. Per specificare la modalità di visualizzazione del nome dell'applicazione sulla console X-Ray, impostate la variabile di `AWS_XRAY_TRACING_NAME` ambiente o la proprietà di `com.amazonaws.xray.strategy.tracingName` sistema. Se non viene fornito alcun nome, viene utilizzato un nome predefinito.
5. Riavvia il server o il contenitore. Le richieste in arrivo e le relative chiamate a valle vengono ora tracciate. Se non vedi i risultati previsti, consulta. the section called “Risoluzione dei problemi”

Configurazione

L'agente X-Ray è configurato da un file JSON esterno fornito dall'utente. Per impostazione predefinita, questo file si trova nella radice del percorso di classe dell'utente (ad esempio, nella sua directory) denominato. `resources` `xray-agent.json` È possibile configurare una posizione personalizzata per il file di configurazione impostando la proprietà `com.amazonaws.xray.configFile` system sul percorso assoluto del file di configurazione.

Di seguito viene mostrato un file di configurazione di esempio.

```
{
    "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
}
```

Specificazione di configurazione

La tabella seguente descrive i valori validi per ogni proprietà. I nomi delle proprietà fanno distinzione tra maiuscole e minuscole, ma le relative chiavi no. Per le proprietà che possono essere sostituite dalle variabili di ambiente e dalle proprietà di sistema, l'ordine di priorità è sempre la variabile di ambiente, quindi la proprietà di sistema e quindi il file di configurazione. Per informazioni sulle proprietà che è possibile sostituire, vedere. Variabili di ambiente Tutti i campi sono facoltativi.

Nome proprietà	Tipo	Valori validi	Description	Variabile di ambiente	Proprietà del sistema	Predefinita
serviceName	Stringa	Qualsiasi stringa	Il nome del servizio strumentato così come verrà visualizzato nella console X-Ray.	AWS_XRAY_NOME_TRACCIAMENTO	com.amazonaws.xray.strategy.tracingName	XRayInstrumentedService
contextMissingStrategy	Stringa	LOG_ERROR, IGNORE_ERROR	L'azione intrapresa dall'agente quando tenta di utilizzare il contesto del segmento X-Ray ma non è presente.	AWS_XRAY_CONTEXT_MISSING	com.amazonaws.xray.strategy.contextMissingStrategy	LOG_ERROR
Indirizzo del demone	Stringa	Indirizzo IP e porta formattati o elenco	L'indirizzo utilizzato dall'agente per	AWS_XRAY_INDIRIZZO_DEMONE	indirizzo com.amazonaws.xray	1270.0.1:2000

Nome proprietà	Tipo	Valori validi	Description	Variabile di ambiente	Proprietà del sistema	Predefinita
		di indirizzi TCP e UDP	comunicare con il demone X-Ray.		.emitter.daemon	
Tracciamento abilitato	Booleano	Vero, falso	Abilita la strumentazione da parte dell'agente X-Ray.	AWS_XRAY_TRACING_ENABLED	com.amazonaws.xray.tracingEnabled	TRUE
Strategia di campionamento	Stringa	CENTRALE, LOCALE, NESSUNO, TUTTO	La strategia di campionamento utilizzata dall'agente. ALL acquisisce tutte le richieste, NONE non acquisisce alcuna richiesta. Vedi le regole di campionamento.	N/D	N/D	CENTRALE

Nome proprietà	Tipo	Valori validi	Description	Variabile di ambiente	Proprietà del sistema	Predefinita
traceIdInjectionprefisso	Stringa	Qualsiasi stringa	Include il prefisso fornito prima della traccia IDs iniettata nei registri.	N/D	N/D	Nessuno (stringa vuota)
samplingRulesManifest	Stringa	Un percorso di file assoluto	Il percorso di un file di regole di campionamento personalizzato da utilizzare come origine delle regole di campionamento per la strategia di campionamento locale o delle regole di fallback per la strategia centrale.	N/D	N/D	DefaultSamplingRules.json

Nome proprietà	Tipo	Valori validi	Description	Variabile di ambiente	Proprietà del sistema	Predefinita
awsServiceHandlerManifesto	Stringa	Un percorso di file assoluto	Il percorso di un elenco di parametri consentiti personalizzato, che acquisisce informazioni aggiuntive dai client AWS SDK.	N/D	N/D	DefaultOperationParameterWhitelist.json
awsSdkVersion	Numero intero	1, 2	Versione dell'AWS SDK for Java che stai utilizzando. Ignorata se non `awsServiceHandlerManifest` è impostata anche questa opzione.	N/D	N/D	2

Nome proprietà	Tipo	Valori validi	Description	Variabile di ambiente	Proprietà del sistema	Predefinita
maxStackTraceLunghezza	Numero intero	Numeri interi non negativi	Il numero massimo di righe di una traccia dello stack da registrare in una traccia.	N/D	N/D	50
Streaming Threshold	Numero intero	Numeri interi non negativi	Dopo aver chiuso almeno questo numero di sottosegmenti, questi vengono trasmessi in streaming al demone per evitare che i blocchi siano troppo out-of-band grandi.	N/D	N/D	100

Nome proprietà	Tipo	Valori validi	Description	Variabile di ambiente	Proprietà del sistema	Predefinita
traceIdInjection	Booleano	Vero, falso	Abilita l'iniezione di X-Ray Trace ID nei log se vengono aggiunte anche le dipendenze e la configurazione descritte nella configurazione di registrazione. Altrimenti, non fa nulla.	N/D	N/D	TRUE
Plugin abilitati	Booleano	Vero, falso	Abilita i plugin che registrano i metadati sugli AWS ambienti in cui operi. Vedi i plugin.	N/D	N/D	TRUE

Nome proprietà	Tipo	Valori validi	Description	Variabile di ambiente	Proprietà del sistema	Predefinita
collectSqlQueries	Booleano	Vero, falso	Registra al meglio le stringhe di query SQL nei sottosegmenti SQL.	N/D	N/D	FALSE
Propagazione del contesto	Booleano	Vero, falso	Se vero, propaga automaticamente il contesto X-Ray tra i thread. Altrimenti, utilizza Thread Local per memorizzare il contesto ed è necessaria la propagazione manuale tra i thread.	N/D	N/D	TRUE

Configurazione della registrazione

Il livello di registro dell'agente X-Ray può essere configurato allo stesso modo dell'X-Ray SDK for Java. Registrazione dei logPer ulteriori informazioni sulla configurazione della registrazione con X-Ray SDK for Java, vedere.

Strumentazione manuale

Se desideri eseguire la strumentazione manuale oltre alla strumentazione automatica dell'agente, aggiungi l'SDK X-Ray come dipendenza dal tuo progetto. Tieni presente che i filtri servlet personalizzati dell'SDK menzionati in Tracing Incoming Requests non sono compatibili con l'agente X-Ray.

> **Note**
>
> È necessario utilizzare la versione più recente di X-Ray SDK per eseguire la strumentazione manuale utilizzando anche l'agente.

Se stai lavorando a un progetto Maven, aggiungi le seguenti dipendenze al tuo file. `pom.xml`

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
    <version>2.11.0</version>
  </dependency>
  </dependencies>
```

Se stai lavorando su un progetto Gradle, aggiungi le seguenti dipendenze al tuo file. `build.gradle`

```
implementation 'com.amazonaws:aws-xray-recorder-sdk-core:2.11.0'
```

Puoi aggiungere sottosegmenti personalizzati oltre alle annotazioni, ai metadati e all'utente IDs durante l'utilizzo dell'agente, proprio come faresti con il normale SDK. L'agente propaga automaticamente il contesto tra i thread, quindi non dovrebbero essere necessarie soluzioni alternative per propagare il contesto quando si lavora con applicazioni multithread.

Risoluzione dei problemi

Poiché l'agente offre una strumentazione completamente automatica, può essere difficile identificare la causa principale di un problema quando si verificano problemi. Se l'agente X-Ray non funziona come previsto, esamina i seguenti problemi e soluzioni. L'agente X-Ray e l'SDK utilizzano Jakarta Commons Logging (JCL). Per visualizzare l'output di registrazione, assicurati che un bridge che collega JCL al tuo backend di registrazione si trovi sul classpath, come nell'esempio seguente: o. `log4j-jcl` `jcl-over-slf4j`

Problema: ho abilitato l'agente Java sulla mia applicazione ma non vedo nulla sulla console X-Ray

Il demone X-Ray è in esecuzione sulla stessa macchina?

In caso contrario, consultate la documentazione del demone X-Ray per configurarlo.

Nei registri delle applicazioni, viene visualizzato un messaggio del tipo «Inizializzazione del registratore dell'agente X-Ray»?

Se l'agente è stato aggiunto correttamente all'applicazione, questo messaggio viene registrato a livello INFO all'avvio dell'applicazione, prima che inizi a ricevere le richieste. Se questo messaggio non è presente, l'agente Java non è in esecuzione con il processo Java. Assicurati di aver seguito correttamente tutti i passaggi di configurazione senza errori di battitura.

Nei registri delle applicazioni, vedi diversi messaggi di errore che dicono qualcosa come «Soppressione dell'eccezione mancante AWS X-Ray nel contesto»?

Questi errori si verificano perché l'agente sta cercando di strumentare le richieste a valle, come le richieste AWS SDK o le query SQL, ma non è riuscito a creare automaticamente un segmento. Se riscontri molti di questi errori, l'agente potrebbe non essere lo strumento migliore per il tuo caso d'uso e potresti prendere in considerazione la strumentazione manuale con X-Ray SDK. In alternativa, puoi abilitare i log di debug di X-Ray SDK per visualizzare la traccia dello stack in cui si verificano le eccezioni mancanti nel contesto. Puoi racchiudere queste parti del codice con segmenti personalizzati, che dovrebbero risolvere questi errori. Per un esempio di avvolgimento delle richieste downstream con segmenti personalizzati, consulta il codice di esempio in Strumenting Startup Code.

Problema: alcuni segmenti che prevedo non vengono visualizzati sulla console X-Ray

La tua applicazione utilizza il multithreading?

Se alcuni segmenti che prevedi di creare non vengono visualizzati nella console, la causa potrebbe essere la presenza di thread in background nell'applicazione. Se la tua applicazione esegue attività

utilizzando thread in background che sono «spara e dimentica», come effettuare una chiamata una tantum a una funzione Lambda con l' AWS SDK o eseguire periodicamente il polling di alcuni endpoint HTTP, ciò potrebbe confondere l'agente mentre propaga il contesto tra i thread. <NAME >Per verificare che questo sia il problema, abilita i log di debug di X-Ray SDK e verifica la presenza di messaggi come: Non emettere segmenti denominati in quanto generano i sottosegmenti in corso. Per ovviare a questo problema, puoi provare a unire i thread in background prima che il server ritorni, per assicurarti che tutto il lavoro svolto in essi venga registrato. In alternativa, puoi impostare la `contextPropagation` configurazione dell'agente in modo da `false` disabilitare la propagazione del contesto nei thread in background. Se lo fai, dovrai strumentare manualmente quei thread con segmenti personalizzati o ignorare le eccezioni mancanti nel contesto che generano.

Hai impostato delle regole di campionamento?

Se sulla console X-Ray compaiono segmenti apparentemente casuali o inaspettati, o se i segmenti che si prevede siano presenti sulla console non lo sono, è possibile che si tratti di un problema di campionamento. L'agente X-Ray applica il campionamento centralizzato a tutti i segmenti che crea, utilizzando le regole della console X-Ray. La regola predefinita prevede il campionamento di 1 segmento al secondo, più il 5% dei segmenti che vengono successivamente campionati. Ciò significa che i segmenti creati rapidamente con l'agente potrebbero non essere campionati. Per risolvere questo problema, è necessario creare regole di campionamento personalizzate sulla console X-Ray che campionino in modo appropriato i segmenti desiderati. Per ulteriori informazioni, vedere campionamento.

Configurazione dell'X-Ray SDK per Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L'X-Ray SDK for Java include una classe denominata `AWSXRay` che fornisce il registratore globale. Questo è un `TracingHandler` che puoi usare per strumentare il tuo codice. Puoi configurare la

registrazione globale per personalizzare il `AWSXRayServletFilter` che crea i segmenti relativi alle chiamate HTTP in entrata.

Sections

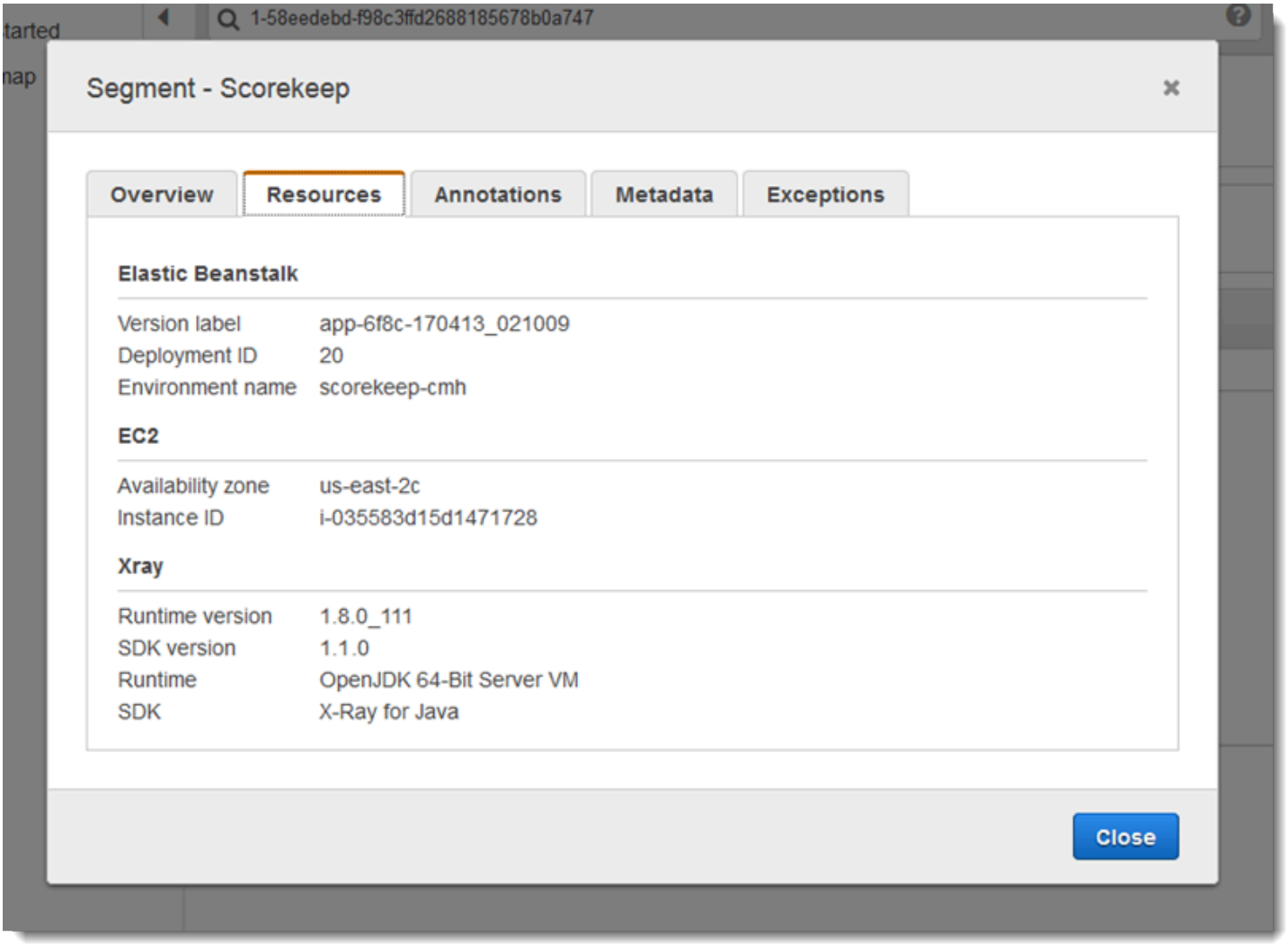
- Plugin di servizio
- Regole di campionamento
- Registrazione dei log
- Listener di segmenti
- Variabili di ambiente
- Proprietà di sistema

Plugin di servizio

`pluginsUtilizzatelo` per registrare informazioni sul servizio che ospita l'applicazione.

Plugin

- Amazon EC2: `EC2Plugin` aggiunge l'ID dell'istanza, la zona di disponibilità e il gruppo di CloudWatch log.
- Elastic `ElasticBeanstalkPlugin` Beanstalk: aggiunge il nome dell'ambiente, l'etichetta della versione e l'ID di distribuzione.
- Amazon ECS: `ECSPlugin` aggiunge l'ID del contenitore.
- Amazon EKS: `EKSPlugin` aggiunge l'ID del contenitore, il nome del cluster, l'ID del pod e il gruppo CloudWatch Logs.



Per usare un plugin, chiama `withPlugin` sul tuo `AWSXRayRecorderBuilder`.

Example src/main/java/scorekeep/WebConfig.java - registratore

```
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());
  }
}
```

L'SDK utilizza anche le impostazioni del plug-in per impostare il `origin` campo sul segmento. Indica il tipo di AWS risorsa che esegue l'applicazione. Quando utilizzi più plugin, l'SDK utilizza il seguente ordine di risoluzione per determinare l'origine: ElasticBeanstalk > EKS > ECS > EC2.

Regole di campionamento

L'SDK utilizza le regole di campionamento definite nella console X-Ray per determinare quali richieste registrare. La regola predefinita tiene traccia della prima richiesta ogni secondo e del cinque percento di eventuali richieste aggiuntive su tutti i servizi che inviano tracce a X-Ray. Crea regole aggiuntive nella console X-Ray per personalizzare la quantità di dati registrati per ciascuna delle tue applicazioni.

L'SDK applica le regole personalizzate nell'ordine in cui sono definite. Se una richiesta corrisponde a più regole personalizzate, l'SDK applica solo la prima regola.

> **Note**
>
> Se l'SDK non riesce a contattare X-Ray per ottenere le regole di campionamento, torna a una regola locale predefinita della prima richiesta ogni secondo e del cinque percento di eventuali richieste aggiuntive per host. Ciò può verificarsi se l'host non dispone dell'autorizzazione per chiamare sampling APIs o non riesce a connettersi al demone X-Ray, che funge da proxy TCP per le chiamate API effettuate dall'SDK.

Puoi anche configurare l'SDK per caricare le regole di campionamento da un documento JSON. L'SDK può utilizzare le regole locali come backup per i casi in cui il campionamento a raggi X non è disponibile o utilizzare esclusivamente regole locali.

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
  }
}
```

Questo esempio definisce una regola personalizzata e una regola predefinita. La regola personalizzata applica una frequenza di campionamento del cinque percento senza alcun numero minimo di richieste da tracciare per i percorsi. `/api/move/` La regola predefinita tiene traccia della prima richiesta ogni secondo e del 10% delle richieste aggiuntive.

Lo svantaggio della definizione locale delle regole è che l'obiettivo fisso viene applicato da ciascuna istanza del registratore in modo indipendente, anziché essere gestito dal servizio X-Ray. Man mano che si installano più host, la tariffa fissa si moltiplica, rendendo più difficile il controllo della quantità di dati registrati.

Sì AWS Lambda, non è possibile modificare la frequenza di campionamento. Se la funzione viene chiamata da un servizio strumentato, le chiamate che hanno generato richieste campionate da quel servizio verranno registrate da Lambda. Se il tracciamento attivo è abilitato e non è presente alcuna intestazione di tracciamento, Lambda prende la decisione di campionamento.

Per fornire le regole di backup in Spring, configura il registratore globale con un codice `CentralizedSamplingStrategy` in una classe di configurazione:

Example src/main/java/myapp/WebConfig.java - configurazione del registratore

```
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());
}
```

Per Tomcat, aggiungi un listener che estenda `ServletContextListener` e registra il listener nel descrittore della distribuzione.

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>
```

Per utilizzare solo regole locali, sostituisci `CentralizedSamplingStrategy` con una `LocalizedSamplingStrategy`.

```
builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));
```

Registrazione dei log

Per impostazione predefinita, l'SDK invia messaggi a `ERROR` livello di file nei log delle applicazioni. È possibile abilitare la registrazione a livello di debug sull'SDK per generare log più dettagliati nel file di registro dell'applicazione. I livelli di registro validi sono`DEBUG`,, e. `INFO` `WARN` `ERROR` `FATAL` `FATAL`Il livello di registro silenzia tutti i messaggi di registro perché l'SDK non esegue la registrazione a un livello irreversibile.

Example application.properties

Imposta il livello di log tramite la proprietà `logging.level.com.amazonaws.xray`.

```
logging.level.com.amazonaws.xray = DEBUG
```

Utilizza i log di debug per identificare i problemi, come ad esempio dei sottosegmenti non chiusi, quando generi dei sottosegmenti manualmente.

Inserimento dell'ID di tracciamento nei log

Per esporre l'ID di tracciamento completo corrente alle istruzioni di log, puoi inserire l'ID nel contesto diagnostico mappato (MDC). Utilizzando l'interfaccia `SegmentListener`, i metodi vengono chiamati dal registratore di X-Ray durante gli eventi del ciclo di vita del segmento. Quando inizia un segmento o un sottosegmento, l'ID di traccia qualificato viene inserito nell'MDC con la chiave `AWS-XRAY-TRACE-ID`. Quando tale segmento termina, la chiave viene rimossa dall'MDC. Questo espone l'ID di tracciamento alla libreria di registrazione in uso. Quando un sottosegmento termina, il relativo ID padre viene inserito nell'MDC.

Example ID di tracciamento completo

L'ID completo è rappresentato come `TraceID@EntityID`

```
1-5df42873-011e96598b447dfca814c156@541b3365be3dafc3
```

Questa funzionalità funziona con le applicazioni Java dotate di AWS X-Ray SDK for Java e supporta le seguenti configurazioni di registrazione:

- SLF4API front-end J con backend Logback
- SLF4API front-end J con backend Log4J2
- API front-end Log4J2 con backend Log4J2

Consulta le seguenti schede per le esigenze di ogni front-end e ogni back-end.

SLF4J Frontend

1. Aggiungi la seguente dipendenza Maven al tuo progetto.

   ```
   <dependency>
       <groupId>com.amazonaws</groupId>
       <artifactId>aws-xray-recorder-sdk-slf4j</artifactId>
       <version>2.11.0</version>
   </dependency>
   ```

2. Includi il metodo `withSegmentListener` durante la costruzione di `AWSXRayRecorder`. Questo aggiunge una `SegmentListener` classe, che inserisce automaticamente una nuova traccia nel J MDC. IDs SLF4

   `SegmentListener` accetta una stringa facoltativa come parametro per configurare il prefisso dell'istruzione di log. Il prefisso può essere configurato nei seguenti modi:

   - Nessuno: utilizza il prefisso predefinito`AWS-XRAY-TRACE-ID`.
   - Vuoto: utilizza una stringa vuota (ad esempio`""`).
   - Personalizzato: utilizza un prefisso personalizzato come definito nella stringa.

   Example Dichiarazione **`AWSXRayRecorderBuilder`**

   ```
   AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
   ```

```
        .standard().withSegmentListener(new SLF4JSegmentListener("CUSTOM-
PREFIX"));
```

Log4J2 front end

1. Aggiungi la seguente dipendenza Maven al tuo progetto.

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-log4j</artifactId>
    <version>2.11.0</version>
</dependency>
```

2. Includi il metodo `withSegmentListener` durante la costruzione di `AWSXRayRecorder`. Ciò aggiungerà una `SegmentListener` classe che inietta automaticamente una nuova traccia completamente qualificata IDs nel SLF4 J MDC.

   `SegmentListener` accetta una stringa facoltativa come parametro per configurare il prefisso dell'istruzione di log. Il prefisso può essere configurato nei seguenti modi:

   - Nessuno: utilizza il prefisso predefinito`AWS-XRAY-TRACE-ID`.
   - Vuoto: utilizza una stringa vuota (ad esempio`""`) e rimuove il prefisso.
   - Personalizzato: utilizza il prefisso personalizzato definito nella stringa.

   Example Dichiarazione **`AWSXRayRecorderBuilder`**

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new Log4JSegmentListener("CUSTOM-
PREFIX"));
```

Logback backend

Per inserire l'ID di tracciamento negli eventi di registro, è necessario modificare il logger `PatternLayout`, che formatta ogni istruzione di registrazione.

1. Trova dove è configurato `patternLayout`. Puoi farlo a livello di codice o tramite un file di configurazione XML. Per ulteriori informazioni, consulta configurazione di Logback.

2. Inserisci `%X{AWS-XRAY-TRACE-ID}` in punto qualsiasi di `patternLayout` per inserire l'ID di tracciamento nelle istruzioni di registrazione future. `%X{}` indica che si sta recuperando un valore con la chiave fornita dall'MDC. Per ulteriori informazioni su PatternLayouts Logback, consulta. PatternLayout

Log4J2 backend

1. Trova dove è configurato `patternLayout`. Puoi eseguire questa operazione a livello di codice o tramite un file di configurazione scritto in formato XML, JSON, YAML o proprietà.

   Per ulteriori informazioni sulla configurazione di Log4J2 tramite un file di configurazione, consulta Configurazione.

   Per ulteriori informazioni sulla configurazione di Log4J2 a livello di codice, consulta Configurazione programmatica.
2. Inserisci `%X{AWS-XRAY-TRACE-ID}` in punto qualsiasi di `PatternLayout` per inserire l'ID di tracciamento nelle istruzioni di registrazione future. `%X{}` indica che si sta recuperando un valore con la chiave fornita dall'MDC. Per ulteriori informazioni su PatternLayouts Log4J2, vedi Pattern Layout.

Esempio di inserimento dell'ID di tracciamento

Di seguito viene mostrata una stringa `PatternLayout` modificata per includere l'ID di tracciamento. L'ID di tracciamento viene stampato dopo il nome del thread (`%t`) e prima del livello di log (`%-5p`).

Example **`PatternLayout`** con inserimento dell'ID

```
%d{HH:mm:ss.SSS} [%t] %X{AWS-XRAY-TRACE-ID} %-5p %m%n
```

AWS X-Ray stampa automaticamente la chiave e l'ID di traccia nell'istruzione di registro per una facile analisi. Di seguito viene illustrata un'istruzione log utilizzando la modifica `PatternLayout`.

Example Istruzione log con inserimento dell'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
```

Il messaggio di registrazione stesso è alloggiato nel modello `%m` e viene impostato quando si chiama il logger.

Listener di segmenti

I segment listener sono un'interfaccia per intercettare eventi del ciclo di vita come l'inizio e la fine dei segmenti prodotti da. `AWSXRayRecorder` L'implementazione di una funzione evento listener di segmenti potrebbe essere quella di aggiungere la stessa annotazione a tutti i sottosegmenti quando vengono creati con `onBeginSubsegment`, registrare un messaggio dopo che ogni segmento è stato inviato al daemon utilizzando `afterEndSegment`, o per registrare le query inviate dagli intercettori SQL che utilizzano `beforeEndSubsegment` per verificare se il sottosegmento rappresenta una query SQL e, in tal caso, aggiungendo ulteriori metadati.

Per visualizzare l'elenco completo delle `SegmentListener` funzioni, consulta la documentazione dell'API AWS X-Ray Recorder SDK for Java.

Nell'esempio seguente viene illustrato come aggiungere un'annotazione coerente a tutti i sottosegmenti durante la creazione con `onBeginSubsegment` e per stampare un messaggio di log alla fine di ogni segmento con `afterEndSegment`.

Example MySegmentListener.java

```
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
import com.amazonaws.xray.listeners.SegmentListener;

public class MySegmentListener implements SegmentListener {
    .....

    @Override
    public void onBeginSubsegment(Subsegment subsegment) {
        subsegment.putAnnotation("annotationKey", "annotationValue");
    }

    @Override
    public void afterEndSegment(Segment segment) {
        // Be mindful not to mutate the segment
        logger.info("Segment with ID " + segment.getId());
    }
}
```

Durante la creazione di `AWSXRayRecorder`, viene fatto quindi riferimento a questo listener.

Example AWSXRayRecorderBuilder dichiarazione

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new MySegmentListener());
```

Variabili di ambiente

È possibile utilizzare le variabili di ambiente per configurare l'X-Ray SDK for Java. L'SDK supporta le seguenti variabili.

- `AWS_XRAY_CONTEXT_MISSING`— Imposta per `RUNTIME_ERROR` generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.

  Valori validi

  - `RUNTIME_ERROR`— Genera un'eccezione di runtime.
  - `LOG_ERROR`— Registra un errore e continua (impostazione predefinita).
  - `IGNORE_ERROR`— Ignora l'errore e continua.

  Gli errori relativi a segmenti o sottosegmenti mancanti possono verificarsi quando si tenta di utilizzare un client con strumenti nel codice di avvio che viene eseguito quando non è aperta alcuna richiesta o nel codice che genera un nuovo thread.
- `AWS_XRAY_DAEMON_ADDRESS`— Imposta l'host e la porta del demone X-Ray. Per impostazione predefinita, l'SDK utilizza `127.0.0.1:2000` sia i dati di traccia (UDP) che il campionamento (TCP). Utilizzate questa variabile se avete configurato il demone per l'ascolto su una porta diversa o se è in esecuzione su un host diverso.

  Formato

  - Stessa porta — *`address`*`:`*`port`*
  - Porte diverse: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_LOG_GROUP`— Imposta il nome di un gruppo di log sul gruppo di log associato all'applicazione. Se il gruppo di log utilizza lo stesso AWS account e la stessa regione dell'applicazione, X-Ray cercherà automaticamente i dati del segmento dell'applicazione utilizzando questo gruppo di log specificato. Per ulteriori informazioni sui gruppi di log, consulta Lavorare con gruppi di log e flussi.

- `AWS_XRAY_TRACING_NAME`— Imposta un nome di servizio che l'SDK utilizza per i segmenti. Sostituisce il nome del servizio impostato sulla strategia di denominazione dei segmenti del filtro servlet.

Le variabili di ambiente sostituiscono i valori equivalenti per le proprietà di sistema e i valori impostati nel codice.

Proprietà di sistema

Puoi utilizzare le proprietà del sistema come alternativa alle variabili d'ambiente per una specifica JVM. L'SDK supporta le seguenti proprietà.

- `com.amazonaws.xray.strategy.tracingName`— Equivalente a. `AWS_XRAY_TRACING_NAME`
- `com.amazonaws.xray.emitters.daemonAddress`— Equivalente a`AWS_XRAY_DAEMON_ADDRESS`.
- `com.amazonaws.xray.strategy.contextMissingStrategy`— Equivalente a`AWS_XRAY_CONTEXT_MISSING`.

Se sono impostate sia una proprietà di sistema che la variabili di ambiente equivalente, sono utilizzati i valori della variabile di ambiente. Entrambi i metodi sostituiscono i valori impostati nel codice.

Tracciamento delle richieste in arrivo con X-Ray SDK for Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi utilizzare l'SDK X-Ray per tracciare le richieste HTTP in entrata che la tua applicazione serve su un'istanza EC2 in Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Utilizza un `Filter` per analizzare le richieste HTTP in entrata. Quando aggiungete il filtro servlet X-Ray all'applicazione, l'X-Ray SDK for Java crea un segmento per ogni richiesta campionata.

Questo segmento include durata, metodo e conclusione della richiesta HTTP. Analisi ulteriori creano sottosegmenti associati a questo segmento.

> **Note**
>
> Per quanto riguarda AWS Lambda le funzioni, Lambda crea un segmento per ogni richiesta campionata. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Ogni segmento ha un nome che identifica l'applicazione nella mappa dei servizi. Il segmento può essere denominato staticamente oppure è possibile configurare l'SDK per denominarlo dinamicamente in base all'intestazione dell'host nella richiesta in entrata. La denominazione dinamica consente di raggruppare le tracce in base al nome di dominio nella richiesta e di applicare un nome predefinito se il nome non corrisponde a uno schema previsto (ad esempio, se l'intestazione dell'host è falsificata).

> **Richieste inoltrate**
>
> Se un sistema di bilanciamento del carico o un altro intermediario inoltra una richiesta all'applicazione, X-Ray prende l'IP del client dall'`X-Forwarded-For`intestazione della richiesta anziché dall'IP di origine nel pacchetto IP. L'IP del client registrato per una richiesta inoltrata può essere falsificato, quindi non dovrebbe essere considerato attendibile.

Quando viene inoltrata una richiesta, l'SDK imposta un campo aggiuntivo nel segmento per indicarlo. Se il segmento contiene il campo `x_forwarded_for` impostato su`true`, l'IP del client è stato preso dall'`X-Forwarded-For`intestazione della richiesta HTTP.

Il gestore dei messaggi crea un segmento per ogni richiesta in entrata con un blocco `http` che contiene le informazioni riportate qui di seguito:

- Metodo HTTP: GET, POST, PUT, DELETE, ecc.
- Indirizzo client: l'indirizzo IP del client che ha inviato la richiesta.
- Codice di risposta: il codice di risposta HTTP per la richiesta completata.
- Tempistica: l'ora di inizio (quando è stata ricevuta la richiesta) e l'ora di fine (quando è stata inviata la risposta).
- Agente utente: il `user-agent` codice della richiesta.

- Lunghezza del contenuto: il `content-length` risultato della risposta.

Sections

- Aggiunta di un filtro di tracciamento all'applicazione (Tomcat)
- Aggiunta di un filtro di tracciamento all'applicazione (Spring)
- Configurazione di una strategia di denominazione dei segmenti

Aggiunta di un filtro di tracciamento all'applicazione (Tomcat)

Per Tomcat, aggiungere un `<filter>` al file `web.xml` del progetto. Utilizza il parametro `fixedName` per specificare un nome di servizio da applicare ai segmenti creati per le richieste in entrata.

Example WEB-INF/web.xml - Tomcat

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.javax.servlet.AWSXRayServletFilter</filter-class>
  <init-param>
    <param-name>fixedName</param-name>
    <param-value>MyApp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>AWSXRayServletFilter</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
```

Aggiunta di un filtro di tracciamento all'applicazione (Spring)

Per Spring, aggiungere un `Filter` alla classe `WebConfig`. Passare il nome del segmento al costruttore `AWSXRayServletFilter` come valore di tipo stringa.

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");
  }
}
```

Configurazione di una strategia di denominazione dei segmenti

AWS X-Ray utilizza un nome di servizio per identificare l'applicazione e distinguerla dalle altre applicazioni, database, risorse esterne APIs e AWS risorse utilizzate dall'applicazione. Quando X-Ray SDK genera segmenti per le richieste in entrata, registra il nome del servizio dell'applicazione nel campo del nome del segmento.

L'X-Ray SDK può denominare i segmenti dopo il nome host nell'intestazione della richiesta HTTP. Tuttavia, questa intestazione può essere falsificata, il che potrebbe causare nodi imprevisti nella mappa dei servizi. Per evitare che l'SDK nomini i segmenti in modo errato a causa di richieste con intestazioni host contraffatte, è necessario specificare un nome predefinito per le richieste in entrata.

Se la tua applicazione soddisfa le richieste per più domini, puoi configurare l'SDK in modo che utilizzi una strategia di denominazione dinamica che rifletta questo aspetto nei nomi dei segmenti. Una strategia di denominazione dinamica consente all'SDK di utilizzare il nome host per le richieste che corrispondono a uno schema previsto e di applicare il nome predefinito alle richieste che non lo fanno.

Ad esempio, potresti avere una singola applicazione che serve le richieste a tre sottodomini:, e. `www.example.com` `api.example.com` `static.example.com` È possibile utilizzare una strategia di denominazione dinamica con lo schema `*.example.com` per identificare i segmenti per ogni sottodominio con un nome diverso, ottenendo tre nodi di servizio sulla mappa dei servizi. Se l'applicazione riceve richieste con un nome host che non corrisponde allo schema, sulla mappa dei servizi verrà visualizzato un quarto nodo con un nome di fallback specificato dall'utente.

Per utilizzare lo stesso nome per tutti i segmenti della richiesta, specifica il nome della tua applicazione quando inizializzi il filtro servlet, come illustrato nella sezione precedente. Ciò ha lo stesso effetto della creazione di una correzione SegmentNamingStrategychiamandola `SegmentNamingStrategy.fixed()` e passandola al `AWSXRayServletFilter`costruttore.

> **Note**
>
> È possibile sovrascrivere il nome di servizio predefinito definito nel codice con la variabile di `AWS_XRAY_TRACING_NAME` ambiente.

Una strategia di denominazione dinamica definisce un modello al quale devono corrispondere i nomi degli host e un nome di default per l'utilizzo qualora il nome dell'host nella richiesta HTTP non corrisponda al modello. Per denominare segmenti in modo dinamico in Tomcat, utilizzare la `dynamicNamingRecognizedHosts` e `dynamicNamingFallbackName` per definire rispettivamente i modelli e il nome di default.

Example WEB-INF/web.xml - Filtro Servlet con assegnazione dinamica dei nomi

```
<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>
```

Per Spring, crea una dinamica SegmentNamingStrategy`SegmentNamingStrategy.dynamic()`chiamando e passala al `AWSXRayServletFilter` costruttore.

Example src/main/java/myapp/WebConfig.java - filtro servlet con denominazione dinamica

```
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"));
  }
}
```

Tracciamento delle chiamate AWS SDK con X-Ray SDK for Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate per Servizi AWS archiviare dati, scrivere in una coda o inviare notifiche, X-Ray SDK for Java tiene traccia delle chiamate downstream in sottosegmenti. Le risorse tracciate Servizi AWS e a cui accedi all'interno di tali servizi (ad esempio, un bucket Amazon S3 o una coda Amazon SQS) vengono visualizzate come nodi downstream sulla mappa di traccia nella console X-Ray.

L'SDK X-Ray per Java strumenta automaticamente tutti i client SDK AWS quando includi i sottomoduli `aws-sdk` e un `aws-sdk-instrumentor` nella build. Se non includi il modulo Instrumentor, puoi scegliere di analizzare alcuni client escludendo gli altri.

Per strumentare i singoli client, rimuovi il `aws-sdk-instrumentor` sottomodulo dalla build e aggiungi un client `XRayClient` as a `TracingHandler` sul tuo AWS SDK utilizzando il client builder del servizio.

Ad esempio, per analizzare un client `AmazonDynamoDB`, passa un gestore del tracciamento `AmazonDynamoDBClientBuilder`.

Example MyModel.java - Client 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();
...
```

Per tutti i servizi, puoi vedere il nome dell'API richiamata nella console X-Ray. Per un sottoinsieme di servizi, l'SDK X-Ray aggiunge informazioni al segmento per fornire una maggiore granularità nella mappa dei servizi.

Ad esempio, quando si effettua una chiamata con un client DynamoDB con strumentazione, l'SDK aggiunge il nome della tabella al segmento per le chiamate destinate a una tabella. Nella console, ogni tabella appare come un nodo separato nella mappa dei servizi, con un nodo DynamoDB generico per le chiamate che non hanno come destinazione una tabella.

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento

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

```
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Quando si accede alle risorse con nome, le chiamate ai seguenti servizi creano ulteriori nodi della mappa del servizio. Le chiamate che non sono hanno come obiettivo risorse specifiche creano un nodo generico per il servizio.

- Amazon DynamoDB: nome della tabella
- Amazon Simple Storage Service: nome del bucket e della chiave
- Amazon Simple Queue Service: nome della coda

Per strumentare le chiamate downstream a Servizi AWS with AWS SDK per Java 2.2 e versioni successive, puoi omettere il `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` modulo dalla configurazione di build. Al suo posto, includi il `aws-xray-recorder-sdk-aws-sdk-v2 module`, quindi analizza i singoli client configurandoli con un `TracingInterceptor`.

Example AWS SDK per Java 2.2 e versioni successive - tracing interceptor

```
import com.amazonaws.xray.interceptors.TracingInterceptor;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
//...
public class MyModel {
private DynamoDbClient client = DynamoDbClient.builder()
.region(Region.US_WEST_2)
.overrideConfiguration(ClientOverrideConfiguration.builder()
.addExecutionInterceptor(new TracingInterceptor())
.build()
)
.build();
//...
```

Tracciamento delle chiamate ai servizi Web HTTP downstream con X-Ray SDK for Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate a microservizi o a HTTP pubblico APIs, è possibile utilizzare la versione `HttpClient` di X-Ray SDK per Java per strumentare tali chiamate e aggiungere l'API al service graph come servizio downstream.

L'X-Ray SDK for Java include `DefaultHttpClient` `HttpClientBuilder` classi che possono essere utilizzate al posto degli HttpComponents equivalenti di Apache per strumentare le chiamate HTTP in uscita.

- `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`

Queste librerie sono contenute nel sottomodulo `aws-xray-recorder-sdk-apache-http`.

È possibile sostituire le istruzioni di importazione esistenti con l'equivalente X-Ray di Instrument all client o utilizzare il nome completo quando si inizializza un client su client specifici dello strumento.

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 si effettua una chiamata a un'API Web downstream, l'X-Ray SDK for Java registra un sottosegmento con informazioni sulla richiesta e sulla risposta HTTP. X-Ray utilizza il sottosegmento per generare un segmento dedotto per l'API remota.

Example Sottosegmento per una chiamata HTTP a valle

```
{
  "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 dedotto per una chiamata HTTP a valle

```
{
  "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
}
```

Tracciamento delle query SQL con X-Ray SDK for Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Interceptori SQL

Strumenta le query del database SQL aggiungendo l'interceptor JDBC X-Ray SDK for Java alla configurazione dell'origine dati.

- PostgreSQL – `com.amazonaws.xray.sql.postgres.TracingInterceptor`

- MySQL – `com.amazonaws.xray.sql.mysql.TracingInterceptor`

Questi collettori si trovano rispettivamente nei sottomoduli `aws-xray-recorder-sql-postgres` e `aws-xray-recorder-sql-mysql`. Implementano `org.apache.tomcat.jdbc.pool.JdbcInterceptor` e sono compatibili con i pool di connessione di Tomcat.

> **Note**
>
> Gli intercettori SQL non registrano la query SQL stessa all'interno dei sottosegmenti per scopi di sicurezza.

Nel caso di Spring, aggiungi il collettore in un file delle proprietà e compila la sorgente dati con il `DataSourceBuilder` di Spring Boot.

Example **`src/main/java/resources/application.properties`** - PostgreSQL JDBC Interceptor

```
spring.datasource.continue-on-error=true
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.jdbc-interceptors=com.amazonaws.xray.sql.postgres.TracingInterceptor
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

Example **`src/main/java/myapp/WebConfig.java`** - Origine dati

```
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();
  }
...
}
```

Per Tomcat, `setJdbcInterceptors` richiamate l'origine dati JDBC con un riferimento alla classe X-Ray SDK for Java.

Example **`src/main/myapp/model.java`** - Origine dati

```
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;");
```

La libreria Tomcat JDBC Data Source è inclusa nell'X-Ray SDK for Java, ma è possibile dichiararla come dipendenza fornita dal documento che la si utilizza.

Example **`pom.xml`**- Origine dati JDBC

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jdbc</artifactId>
  <version>8.0.36</version>
  <scope>provided</scope>
</dependency>
```

Decoratore di tracciamento SQL nativo

- Aggiungilo `aws-xray-recorder-sdk-sql` alle tue dipendenze.
- Decora l'origine dati, la connessione o la dichiarazione del database.

```
dataSource = TracingDataSource.decorate(dataSource)
connection = TracingConnection.decorate(connection)
statement = TracingStatement.decorateStatement(statement)
preparedStatement = TracingStatement.decoratePreparedStatement(preparedStatement,
 sql)
callableStatement = TracingStatement.decorateCallableStatement(callableStatement,
 sql)
```

Generazione di sottosegmenti personalizzati con l' X-Ray SDK for Java

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità di manutenzione, in cui AWS verranno limitate le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

I sottosegmenti estendono il segmento di una traccia con dettagli sul lavoro svolto per soddisfare una richiesta. Ogni volta che effettui una chiamata con un client dotato di strumenti, l' X-Ray SDK registra le informazioni generate in un sottosegmento. Puoi creare sottosegmenti aggiuntivi per raggruppare altri sottosegmenti, misurare le prestazioni di una sezione di codice o registrare annotazioni e metadati.

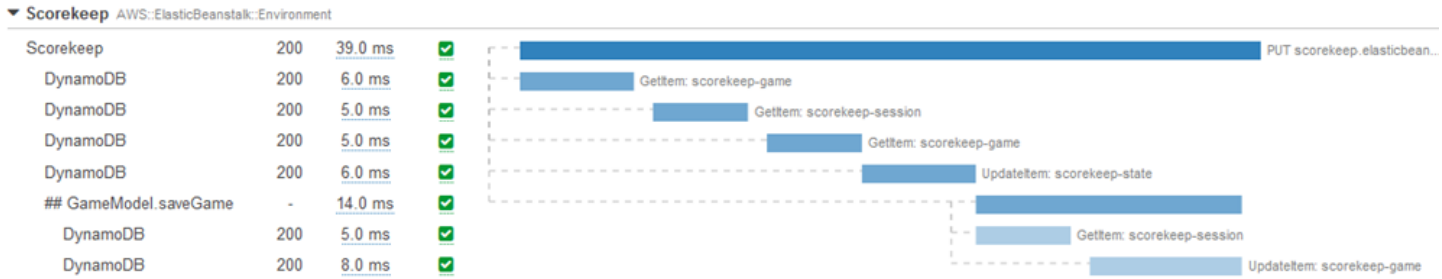
Per gestire i sottosegmenti, utilizza i metodi `beginSubsegment` e `endSubsegment`.

Example GameModel.java - sottosegmento personalizzato

```
import com.amazonaws.xray.AWSXRay;
...
```

```
public void saveGame(Game game) throws SessionNotFoundException {
  // wrap in subsegment
  Subsegment subsegment = AWSXRay.beginSubsegment("Save Game");
  try {
    // check session
    String sessionId = game.getSession();
    if (sessionModel.loadSession(sessionId) == null ) {
      throw new SessionNotFoundException(sessionId);
    }
    mapper.save(game);
  } catch (Exception e) {
    subsegment.addException(e);
    throw e;
  } finally {
    AWSXRay.endSubsegment();
  }
}
```

In questo esempio, il codice all'interno del sottosegmento carica la sessione del gioco da DynamoDB con un metodo sul modello di sessione e utilizza il mapper DynamoDB per salvare il AWS SDK per Java gioco. Inserendo questo codice in un sottosegmento, le chiamate DynamoDB sono figli del `Save Game` sottosegmento nella vista di traccia della console.



Se il codice nel tuo sottosegmento genera eccezioni gestite, eseguine il wrapping in un blocco `try` e chiama `AWSXRay.endSubsegment()` in un blocco `finally` per assicurare che il sottosegmento sia sempre chiuso. Se un sottosegmento non è chiuso, il segmento principale non può essere completato e non verrà inviato a. X-Ray

Per il codice che non genera eccezioni controllate, puoi passare il codice a `AWSXRay.CreateSubsegment` come funzione Lambda.

Example Funzione Lambda del sottosegmento

```
import com.amazonaws.xray.AWSXRay;
```

```
AWSXRay.createSubsegment("getMovies", (subsegment) -> {
    // function code
});
```

Quando crei un sottosegmento all'interno di un segmento o di un altro sottosegmento, l' X-Ray SDK for Java genera un ID per esso e registra l'ora di inizio e l'ora di fine.

Example Sottosegmento con metadati

```
"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"
    }
  },
```

Per la programmazione asincrona e multithread, è necessario passare manualmente il sottosegmento al `endSubsegment()` metodo per assicurarvi che sia chiuso correttamente, poiché il contesto può essere modificato durante l'esecuzione asincrona. X-Ray Se un sottosegmento asincrono viene chiuso dopo la chiusura del segmento principale, questo metodo trasmetterà automaticamente l'intero segmento al demone. X-Ray

Example Sottosegmento asincrono

```
@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();
}
```

Aggiungi annotazioni e metadati ai segmenti con X-Ray SDK for Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile registrare informazioni aggiuntive sulle richieste, sull'ambiente o sull'applicazione con annotazioni e metadati. È possibile aggiungere annotazioni e metadati ai segmenti creati da X-Ray SDK o ai sottosegmenti personalizzati creati dall'utente.

Le annotazioni sono coppie chiave-valore con stringhe, numeri o valori booleani. Le annotazioni sono indicizzate per essere utilizzate con le espressioni di filtro. Utilizzale per registrare i dati che desideri utilizzare per raggruppare le tracce nella console oppure per chiamare l'API `GetTraceSummaries`.

I metadati sono coppie chiave-valore che possono avere valori di qualsiasi tipo, inclusi oggetti ed elenchi, ma non sono indicizzati per essere utilizzati con le espressioni di filtro. Utilizzate i metadati per registrare dati aggiuntivi che desiderate archiviare nella traccia ma che non è necessario utilizzare con la ricerca.

Oltre ad annotazioni e metadati, sui segmenti puoi anche registrare le stringhe degli ID utente. IDs Gli utenti vengono registrati in un campo separato sui segmenti e indicizzati per essere utilizzati nella ricerca.

Sections

- Registrazione delle annotazioni con X-Ray SDK for Java
- Registrazione di metadati con X-Ray SDK for Java
- Utente che registra IDs con X-Ray SDK for Java

Registrazione delle annotazioni con X-Ray SDK for Java

Utilizza le annotazioni per memorizzare le informazioni su segmenti o sottosegmenti che desideri siano indicizzate per la ricerca.

Requisiti per le annotazioni

- Chiavi: la chiave per un'annotazione a raggi X può contenere fino a 500 caratteri alfanumerici. Non è possibile utilizzare spazi o simboli diversi da un punto o un punto (.)
- Valori: il valore di un'annotazione X-Ray può contenere fino a 1.000 caratteri Unicode.
- Il numero di annotazioni: è possibile utilizzare fino a 50 annotazioni per traccia.

Per registrare le annotazioni

1. Ottenere un riferimento al segmento o sottosegmento corrente da `AWSXRay`.

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

   or

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Subsegment;
   ...
   Subsegment document = AWSXRay.getCurrentSubsegment();
   ```

2. Chiamare `putAnnotation` con una chiave di tipo Stringa e un valore booleano, numerico o di tipo Stringa.

   ```
   document.putAnnotation("mykey", "my value");
   ```

   L'esempio seguente mostra come effettuare una chiamata `putAnnotation` con una chiave String che include un punto e un valore booleano, numerico o stringa.

   ```
   document.putAnnotation("testkey.test", "my value");
   ```

L'SDK memorizza le annotazioni come coppie chiave-valore in un oggetto `annotations` all'interno del documento di segmento. Se chiami `putAnnotation` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Per trovare tracciamenti con annotazioni contenenti valori specifici, utilizza la parola chiave `annotation[key]` in un'espressione filtro.

Example **src/main/java/scorekeep/GameModel.java**— Annotazioni e metadati

```
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();
    }
  }
```

Registrazione di metadati con X-Ray SDK for Java

Utilizza i metadati per memorizzare le informazioni su segmenti o sottosegmenti che non è necessario che siano indicizzate per la ricerca. I valori dei metadati possono essere stringhe, numeri, valori booleani o qualsiasi oggetto che possa essere serializzato in un oggetto o in un vettore JSON.

Per registrare i metadati

1. Ottenere un riferimento al segmento o sottosegmento corrente da `AWSXRay`.

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

   or

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Subsegment;
   ...
   Subsegment document = AWSXRay.getCurrentSubsegment();
   ```

2. Chiamare `putMetadata` con un namespace di tipo stringa, una chiave di tipo Stringa e un valore booleano, numerico, di tipo Stringa o di tipo oggetto.

   ```
   document.putMetadata("my namespace", "my key", "my value");
   ```

   or

   Invocare `putMetadata` semplicemente con una chiave e un valore.

   ```
   document.putMetadata("my key", "my value");
   ```

Se non specifichi un namespace, l'SDK utilizza `default`. Se chiami `putMetadata` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Example **src/main/java/scorekeep/GameModel.java**— Annotazioni e metadati

```
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();
    }
  }
```

Utente che registra IDs con X-Ray SDK for Java

Registra i segmenti dell'utente IDs su richiesta per identificare l'utente che ha inviato la richiesta.

Per registrare l'utente IDs

1. Ottenere un riferimento al segmento corrente da `AWSXRay`.

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

2. Chiamare `setUser` con una stringa che rappresenta l'ID dell'utente che ha inviato la richiesta.

   ```
   document.setUser("U12345");
   ```

Puoi chiamare `setUser` nel tuo controller per registrare l'ID utente non appena l'applicazione inizia ad elaborare una richiesta. Se usi il segmento solo per impostare l'ID utente, puoi concatenare le chiamate in un'unica riga.

Example src/main/java/scorekeep/MoveController.java — ID utente

```
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);
  }
```

Per trovare tracciamenti associati ad un ID utente, utilizza la parola chiave `user` in un'espressione filtro.

AWS X-Ray metriche per X-Ray SDK for Java

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Questo argomento descrive lo spazio dei nomi, le metriche e le dimensioni. AWS X-Ray Puoi utilizzare X-Ray SDK for Java per pubblicare metriche CloudWatch Amazon non campionate dai segmenti X-Ray raccolti. Questi parametri sono derivati dall'ora di inizio e di fine del segmento e dai flag di stato di errore, errore e limitazione. Utilizzare questi parametri di tracciamento per esporre tentativi e problemi di dipendenza all'interno di sottosegmenti.

CloudWatch è un archivio di metriche. Una metrica è il concetto fondamentale CloudWatch e rappresenta un insieme di punti dati ordinati nel tempo. Pubblichi (o Servizi AWS) punti dati delle metriche CloudWatch e recuperi le statistiche su tali punti dati come un insieme ordinato di dati di serie temporali.

I parametri sono definiti in modo univoco da un nome, un namespace e una o più dimensioni. A ogni punto dati è associato un timestamp e, facoltativamente, un'unità di misura. Quando si richiedono statistiche, il flusso di dati restituito viene identificato da namespace, nome parametro e dimensione.

Per ulteriori informazioni CloudWatch, consulta la Amazon CloudWatch User Guide.

Metriche X-Ray CloudWatch

Lo spazio dei nomi `ServiceMetrics/SDK` include le metriche descritte di seguito.

Metrica	Statistiche disponibili	Description	unità
`Latency`	Media, Minimo, Massimo, Conteggio	La differenza tra l'ora di inizio e di fine. Media, minima e massima descrivono la latenza operativa. Il conteggio descrive il conteggio delle chiamate.	Millisecondi
`ErrorRate`	Average (Media), Sum (Somma)	La percentuale di richieste non riuscite con un codice di stato `4xx Client Error`, causando un errore.	Percentuale
`FaultRate`	Average (Media), Sum (Somma)	Il tasso di tracciamenti non riusciti con un codice di stato `5xx Server Error`, causando un errore.	Percentuale
`ThrottleRate`	Average (Media), Sum (Somma)	La velocità di tracciamento limitate che restituiscono un codice di stato 429. Questo è un sottoinsi	Percentuale

Metrica	Statistiche disponibili	Description	unità
		eme della metrica `ErrorRate` .	
`OkRate`	Average (Media), Sum (Somma)	Il tasso di richieste tracciate risultante in un codice di stato OK.	Percentuale

Dimensioni dei raggi X CloudWatch

Utilizzate le dimensioni nella tabella seguente per rifinire le metriche restituite per le vostre applicazioni con strumentazione a raggi X. Java

Dimensione	Description
`ServiceType`	Il tipo di servizio, ad esempio, `AWS::EC2::Instance` o `NONE`, se non noto.
`ServiceName`	Il nome canonico del servizio.

Abilita le metriche X-Ray CloudWatch

Utilizzate la seguente procedura per abilitare le metriche di traccia nella vostra applicazione strumentata. Java

Per configurare i parametri di tracciamento

1. Aggiungi il `aws-xray-recorder-sdk-metrics` pacchetto come dipendenza. Apache Maven Per ulteriori informazioni, vedere X-Ray SDK for Java Submodules.
2. Attiva un nuovo `MetricsSegmentListener()` come parte della build del registratore globale.

   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. Implementa l' CloudWatch agente per raccogliere i parametri utilizzando Amazon Elastic Compute Cloud (Amazon) EC2, Amazon Elastic Container Service (Amazon ECS) o Amazon Elastic Kubernetes Service (Amazon EKS):

   - Per configurare Amazon EC2, consulta Installazione dell' CloudWatch agente.
   - Per configurare Amazon ECS, consulta Monitorare i contenitori Amazon ECS utilizzando Container Insights.
   - Per configurare Amazon EKS, consulta Installare l' CloudWatch agente utilizzando il componente aggiuntivo Amazon CloudWatch Observability EKS.

4. Configura l'SDK per comunicare con l'agente. CloudWatch Per impostazione predefinita, l'SDK comunica con l' CloudWatch agente sull'indirizzo. `127.0.0.1` Puoi configurare indirizzi alternativi impostando la variabile di ambiente o la proprietà Java su `address:port`.

   Example Variabile di ambiente

   ```
   AWS_XRAY_METRICS_DAEMON_ADDRESS=address:port
   ```

Example Proprietà Java

```
com.amazonaws.xray.metrics.daemonAddress=address:port
```

Per convalidare la configurazione

1. Accedi a Console di gestione AWS e apri la CloudWatch console all'indirizzo. https://console.aws.amazon.com/cloudwatch/
2. Apri la scheda Metrics (Parametri) per osservare l'afflusso dei tuoi parametri.
3. (Facoltativo) Nella CloudWatch console, nella scheda Registri, apri il gruppo di `ServiceMetricsSDK` log. Cerca un flusso di log che corrisponda ai parametri host e conferma i messaggi di registro.

Passaggio del contesto del segmento tra thread in un'applicazione multithread

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando crei un nuovo thread nella tua applicazione, l'`AWSXRayRecorder` non mantenere un riferimento all'Entità del segmento o sottosegmento correnti. Se utilizzi un client dotato di strumenti nel nuovo thread, l'SDK tenta di scrivere su un segmento che non esiste, causando un. SegmentNotFoundException

Per evitare di generare eccezioni durante lo sviluppo, puoi configurare il registratore con un comando ContextMissingStrategyche gli indichi invece di registrare un errore. È possibile configurare la strategia in codice con SetContextMissingStrategyo configurare opzioni equivalenti con una variabile di ambiente o una proprietà di sistema.

Un modo di risolvere l'errore è utilizzare un nuovo segmento chiamando beginSegment quando inizia il thread e endSegment quando viene chiuso. Questo funziona se si sta analizzando una porzione di codice che non viene eseguita in risposta ad una richiesta HTTP, come il codice eseguito all'avvio dell'applicazione.

Se utilizzi più thread per gestire le richieste in entrata, puoi passare il segmento o il sottosegmento correnti al nuovo thread e fornirli al registratore globale. In questo modo le informazioni registrate all'interno del nuovo thread sono associate allo stesso segmento delle altre informazioni registrate sulla richiesta. Una volta che il segmento è disponibile nel nuovo thread, puoi eseguire qualsiasi eseguibile con accesso al contesto di quel segmento utilizzando il `segment.run(() -> { ... })` metodo.

Consulta Utilizzo dei client analizzati nei thread worker per un esempio.

Utilizzo di X-Ray con programmazione asincrona

L'X-Ray SDK for Java può essere utilizzato in programmi Java asincroni con. SegmentContextExecutors SegmentContextExecutor Implementa l'interfaccia Executor, il che significa che può essere passata a tutte le operazioni asincrone di un. CompletableFuture Ciò garantisce che qualsiasi operazione asincrona venga eseguita con il segmento corretto nel suo contesto.

Example Esempio App.java: passaggio a SegmentContextExecutor CompletableFuture

```
DynamoDbAsyncClient client = DynamoDbAsyncClient.create();

AWSXRay.beginSegment();

// ...

client.getItem(request).thenComposeAsync(response -> {
    // If we did not provide the segment context executor, this request would not be
 traced correctly.
    return client.getItem(request2);
}, SegmentContextExecutors.newSegmentContextExecutor());
```

AOP con Spring e X-Ray SDK for Java

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026 AWS X-Ray SDKs/Daemon entrerà in modalità manutenzione, in cui AWS verranno limitate le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

Questo argomento descrive come utilizzare l' X-Ray SDK e lo Spring Framework per strumentare l'applicazione senza modificarne la logica di base. Ciò significa che ora esiste un modo non invasivo per strumentare le applicazioni in esecuzione in remoto. AWS

Per abilitare AOP in Spring

1. Configurare Spring
2. Aggiungi un filtro di tracciamento alla tua applicazione
3. Annotare il codice o implementare un'interfaccia
4. X-Ray Attivalo nella tua applicazione

Configurazione di Spring

Per configurare Spring affinché utilizzi AOP per l'analisi della tua applicazione, puoi utilizzare Maven o Gradle.

Se utilizzi Maven per compilare l'applicazione, aggiungi la seguente dipendenza nel file `pom.xml`.

```
<dependency>
     <groupId>com.amazonaws</groupId>
     <artifactId>aws-xray-recorder-sdk-spring</artifactId>
     <version>2.11.0</version>
</dependency>
```

Se utilizzi Gradle, aggiungi la seguente dipendenze nel file `build.gradle`.

```
compile 'com.amazonaws:aws-xray-recorder-sdk-spring:2.11.0'
```

Configurazione di Spring Boot

Oltre alla dipendenza Spring descritta nella sezione precedente, se stai usando Spring Boot, aggiungi la seguente dipendenza se non è già nel tuo 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'
```

Aggiungere un filtro di tracciamento all'applicazione

Aggiungi un `Filter` alla tua `WebConfig` classe. Passare il nome del segmento al costruttore `AWSXRayServletFilter` come valore di tipo stringa. Per ulteriori informazioni sul tracciamento dei filtri e sulla strumentazione delle richieste in arrivo, consulta. Tracciamento delle richieste in arrivo con X-Ray SDK for Java

Example src/main//- primavera java/myapp WebConfig.java

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
```

```
  }
}
```

Jakarta Support

Spring 6 utilizza Jakarta anziché Javax per la sua Enterprise Edition. Per supportare questo nuovo spazio dei nomi, X-Ray ha creato un set parallelo di classi che vivono nel proprio spazio dei nomi di Jakarta.

Per le classi di filtro, sostituisci con. `javax` `jakarta` Quando configurate una strategia di denominazione dei segmenti, aggiungete `jakarta` prima del nome della classe di strategia di denominazione, come nell'esempio seguente:

```
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"));
    }
}
```

Annotazione del codice o implementazione di un'interfaccia

Le tue classi devono essere annotate con l'`@XRayEnabled`annotazione o implementare l'interfaccia. `XRayTraced` Questo dice al sistema AOP di racchiudere le funzioni della classe interessata per la strumentazione. X-Ray

X-Ray Attivazione nella tua applicazione

Per attivare il X-Ray tracciamento nell'applicazione, il codice deve estendere la classe astratta `BaseAbstractXRayInterceptor` sovrascrivendo i seguenti metodi.

- `generateMetadata`—Questa funzione consente la personalizzazione dei metadati allegati alla traccia della funzione corrente. Per impostazione predefinita, il nome della classe della funzione in

esecuzione viene registrato nei metadati. Puoi aggiungere altri dati se hai bisogno di informazioni aggiuntive.

- `xrayEnabledClasses`—Questa funzione è vuota e dovrebbe rimanere tale. Serve come host per un pointcut che indica ai collettori di quali metodi effettuare il wrapping. Definisci il pointcut specificando quali delle classi che annotate con `@XRayEnabled` tracciare. La seguente istruzione pointcut indica al collettore di eseguire il wrapping di tutti i bean controller annotati con l'annotazione `@XRayEnabled`.

```
@Pointcut(“@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)”)
```

Se il tuo progetto utilizza Spring Data JPA, valuta la possibilità di estendere from `AbstractXRayInterceptor` anziché. `BaseAbstractXRayInterceptor`

Esempio

Il codice seguente estende la classe astratta. `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() {}

}
```

Il codice seguente è una classe che verrà strumentata da X-Ray.

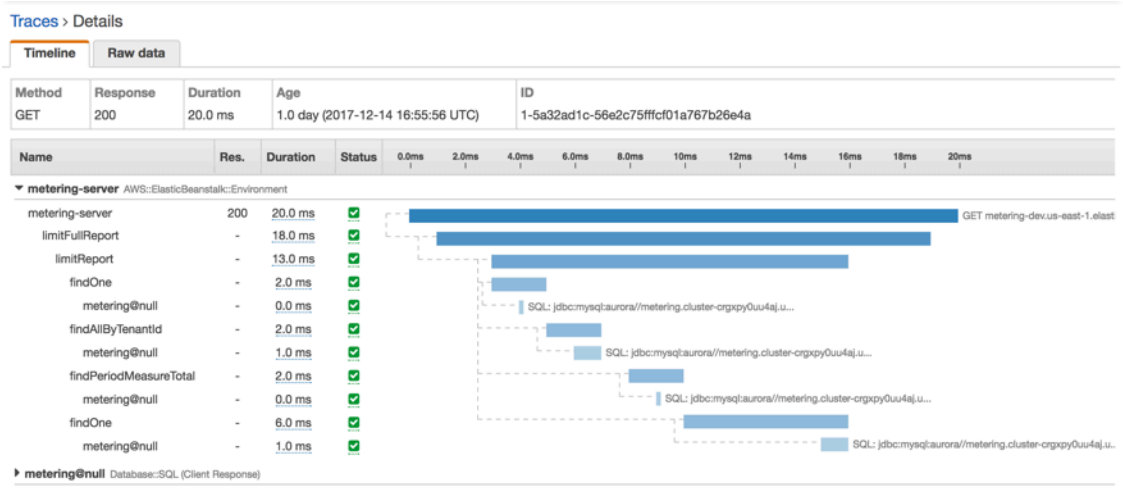
```
@Service
@XRayEnabled
public class MyServiceImpl implements MyService {
    private final MyEntityRepository myEntityRepository;
```

```
    @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 hai configurato la tua applicazione correttamente, dovresti vedere l'intero stack di chiamate dell'applicazione, dai controller fino alle chiamate al servizio, come illustrato nella seguente schermata della console.



Utilizzo di Node.js

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Esistono due modi per utilizzare l'applicazione Node.js per inviare tracce a X-Ray:

- AWS Distro for OpenTelemetry JavaScript: una AWS distribuzione che fornisce un set di librerie open source per l'invio di metriche e tracce correlate a più soluzioni di AWS monitoraggio, tra cui Amazon AWS X-Ray e Amazon OpenSearch Service CloudWatch, tramite AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK per Node.js — Un set di librerie per la generazione e l'invio di tracce a X-Ray tramite il demone X-Ray.

Per ulteriori informazioni, consulta Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs.

AWS Distro per OpenTelemetry JavaScript

Con AWS Distro for OpenTelemetry (ADOT) JavaScript, puoi strumentare le tue applicazioni una sola volta e inviare metriche e tracce correlate a più AWS soluzioni di monitoraggio tra cui Amazon AWS X-Ray e Amazon CloudWatch Service. OpenSearch L'utilizzo di X-Ray con AWS Distro for OpenTelemetry richiede due componenti: un OpenTelemetry SDK abilitato per l'uso con X-Ray e AWS Distro for OpenTelemetry Collector abilitato per l'uso con X-Ray.

Per iniziare, consulta la Distro per la documentazione.AWS OpenTelemetry JavaScript

> **Note**
>
> ADOT JavaScript è supportato per tutte le applicazioni Node.js lato server. ADOT non JavaScript è in grado di esportare dati in X-Ray dai client browser.

Per ulteriori informazioni sull'utilizzo di AWS Distro for OpenTelemetry with AWS X-Ray e altro Servizi AWS, consultate Distro for OpenTelemetry o AWS Distro for Documentation.AWS OpenTelemetry

Per ulteriori informazioni sul supporto e l'utilizzo della lingua, vedere AWS Observability on. GitHub

AWS X-Ray SDK per Node.js

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L'SDK X-Ray per Node.js è una libreria per le applicazioni Web Express e le funzioni Node.js Lambda che fornisce classi e metodi per generare e inviare dati di traccia al demone X-Ray. I dati di traccia includono informazioni sulle richieste HTTP in entrata servite dall'applicazione e sulle chiamate effettuate dall'applicazione ai servizi downstream utilizzando i client SDK o HTTP. AWS

> **Note**
>
> L'X-Ray SDK per Node.js è un progetto open source supportato per le versioni 14.x e successive di Node.js. Puoi seguire il progetto e inviare problemi e richieste pull su GitHub: github. com/aws/aws- xray-sdk-node

Se utilizzi Express, per tracciare le richieste in entrata inizia aggiungendo l'SDK come middleware sul tuo application server. Il middleware crea un segmento per ogni richiesta tracciata e completa

il segmento quando viene inviata la risposta. Fino a che il segmento è aperto puoi usare i metodi del client dell'SDK per aggiungere informazioni al segmento e creare sottosegmenti per tracciare le chiamate a valle. L'SDK, inoltre, registra automaticamente le eccezioni sollevate dall'applicazione per il tempo durante il quale il segmento è aperto.

Per le funzioni Lambda richiamate da un'applicazione o un servizio strumentato, Lambda legge l'intestazione di tracciamento e traccia automaticamente le richieste campionate. Per altre funzioni, puoi configurare Lambda per campionare e tracciare le richieste in arrivo. In entrambi i casi, Lambda crea il segmento e lo fornisce all'X-Ray SDK.

> **Note**
>
> Su Lambda, l'SDK X-Ray è opzionale. Se non lo usi nella tua funzione, la mappa dei servizi includerà comunque un nodo per il servizio Lambda e uno per ogni funzione Lambda. Aggiungendo l'SDK, puoi utilizzare il codice della funzione per aggiungere sottosegmenti al segmento di funzione registrato da Lambda. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Successivamente, utilizzate l'X-Ray SDK per Node.js per strumentare il vostro AWS SDK nei client Node.js. JavaScript Ogni volta che effettui una chiamata a un downstream Servizio AWS o a una risorsa con un client dotato di strumenti, l'SDK registra le informazioni sulla chiamata in un sottosegmento. Servizi AWS e le risorse a cui accedi all'interno dei servizi vengono visualizzate come nodi a valle sulla mappa di traccia per aiutarti a identificare errori e problemi di limitazione sulle singole connessioni.

L'X-Ray SDK per Node.js fornisce anche la strumentazione per le chiamate downstream a query HTTP Web e SQL. APIs Per memorizzare le informazioni sulle chiamate HTTP in uscita includi il client HTTP nel metodo di tracciamento dell'SDK. Nel caso di client SQL, utilizza il metodo di tracciamento per il tuo tipo di database.

Pr determinare quali richieste tracciare, il middleware applica le regole di campionamento alle richieste in entrata. È possibile configurare X-Ray SDK per Node.js per regolare il comportamento di campionamento o per registrare informazioni sulle risorse di AWS calcolo su cui viene eseguita l'applicazione.

Registra ulteriori informazioni sulle richieste e sull'attività svolta dalla tua applicazione in annotazioni e metadati. Le annotazioni sono semplici coppie chiave-valore indicizzati per l'uso con espressioni

filtro, in modo da poter cercare tracce che contengono dati specifici. Le immissioni di metadati sono meno restrittive e possono registrare interi oggetti e matrici, tutto ciò che può essere serializzato in JSON.

> **Annotazioni e metadata**
>
> Le annotazioni e i metadati sono testo arbitrario che aggiungi ai segmenti con X-Ray SDK. Le annotazioni vengono indicizzate per essere utilizzate con le espressioni di filtro. I metadati non sono indicizzati, ma possono essere visualizzati nel segmento non elaborato con la console X-Ray o l'API. Chiunque conceda l'accesso in lettura a X-Ray può visualizzare questi dati.

Quando disponi di una notevole quantità di client analizzati nel tuo codice, un singolo segmento per la richiesta può contenere un numero elevato di sottosegmenti, uno per ciascuna delle chiamate effettuate con un client analizzato. Puoi organizzare e raggruppare i sottosegmenti inglobando le chiamate del client sottosegmenti personalizzati. Puoi creare un sottosegmento personalizzato per un'intera funzione o qualsiasi porzione di codice, e memorizzare metadati e annotazioni sul sottosegmento invece di scrivere tutto all'interno del segmento padre.

Per la documentazione di riferimento sulle classi e i metodi dell'SDK, consulta l'AWS X-Ray SDK for Node.js API Reference.

Requisiti

L'X-Ray SDK per Node.js richiede Node.js e le seguenti librerie:

- `atomic-batcher`— 1.0.2
- `cls-hooked`— 4.2.2
- `pkginfo`— 0.4.0
- `semver`— 5.3.0

L'SDK preleva queste librerie durante la sua installazione con NPM.

Per tracciare i client AWS SDK, X-Ray SDK per Node.js richiede una versione minima AWS dell' JavaScript SDK per Node.js.

- `aws-sdk`— 2.7.15

Gestione delle dipendenze

L'X-Ray SDK per Node.js è disponibile presso NPM.

- Package — aws-xray-sdk

Per lo sviluppo in locale, installa l'SDK nella cartella del tuo progetto con npm.

```
~/nodejs-xray$ npm install aws-xray-sdk
aws-xray-sdk@3.3.3
  ### aws-xray-sdk-core@3.3.3
  # ### @aws-sdk/service-error-classification@3.15.0
  # ### @aws-sdk/types@3.15.0
  # ### @types/cls-hooked@4.3.3
  # # ### @types/node@15.3.0
  # ### atomic-batcher@1.0.2
  # ### cls-hooked@4.2.2
  # # ### async-hook-jl@1.7.6
  # # # ### stack-chain@1.3.7
  # # ### emitter-listener@1.1.2
  # #   ### shimmer@1.2.1
  # ### semver@5.7.1
  ### aws-xray-sdk-express@3.3.3
  ### aws-xray-sdk-mysql@3.3.3
  ### aws-xray-sdk-postgres@3.3.3
```

Utilizza l'opzione `--save` per salvare l'SDK come dipendenza nel file `package.json` della tua applicazione.

```
~/nodejs-xray$ npm install aws-xray-sdk --save
aws-xray-sdk@3.3.3
```

Se l'applicazione presenta dipendenze le cui versioni sono in conflitto con le dipendenze dell'SDK X-Ray, verranno installate entrambe le versioni per garantire la compatibilità. Per maggiori dettagli, consulta la documentazione ufficiale di NPM per la risoluzione delle dipendenze.

Esempi di Node.js

Utilizza l' AWS X-Ray SDK per Node.js per avere una end-to-end visione delle richieste mentre viaggiano attraverso le tue applicazioni Node.js.

- Applicazione di esempio Node.js attiva. GitHub

Configurazione dell'SDK X-Ray per Node.js

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile configurare X-Ray SDK per Node.js con plug-in per includere informazioni sul servizio su cui viene eseguita l'applicazione, modificare il comportamento di campionamento predefinito o aggiungere regole di campionamento che si applicano alle richieste a percorsi specifici.

Sections

- Plugin di servizio
- Regole di campionamento
- Registrazione dei log
- Indirizzo del demone X-Ray
- Variabili di ambiente

Plugin di servizio

Utilizzatelo `plugins` per registrare informazioni sul servizio che ospita l'applicazione.

Plugin

- Amazon EC2 : `EC2Plugin` aggiunge l'ID dell'istanza, la zona di disponibilità e il gruppo di CloudWatch log.
- Elastic `ElasticBeanstalkPlugin` Beanstalk: aggiunge il nome dell'ambiente, l'etichetta della versione e l'ID di distribuzione.
- Amazon ECS: `ECSPlugin` aggiunge l'ID del contenitore.

Per utilizzare un plug-in, configura il client X-Ray SDK per Node.js utilizzando il metodo. `config`

Example app.js - Plugin

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.config([AWSXRay.plugins.EC2Plugin,AWSXRay.plugins.ElasticBeanstalkPlugin]);
```

L'SDK utilizza anche le impostazioni del plug-in per impostare il `origin` campo sul segmento. Indica il tipo di AWS risorsa che esegue l'applicazione. Quando si utilizzano più plug-in, l'SDK utilizza il seguente ordine di risoluzione per determinare l'origine: ElasticBeanstalk > EKS > ECS >. EC2

Regole di campionamento

L'SDK utilizza le regole di campionamento definite nella console X-Ray per determinare quali richieste registrare. La regola predefinita tiene traccia della prima richiesta ogni secondo e del cinque percento di eventuali richieste aggiuntive su tutti i servizi che inviano tracce a X-Ray. Crea regole aggiuntive nella console X-Ray per personalizzare la quantità di dati registrati per ciascuna delle tue applicazioni.

L'SDK applica le regole personalizzate nell'ordine in cui sono definite. Se una richiesta corrisponde a più regole personalizzate, l'SDK applica solo la prima regola.

> **Note**
>
> Se l'SDK non riesce a contattare X-Ray per ottenere le regole di campionamento, torna a una regola locale predefinita della prima richiesta ogni secondo e del cinque percento di eventuali richieste aggiuntive per host. Ciò può verificarsi se l'host non dispone dell'autorizzazione per chiamare sampling APIs o non riesce a connettersi al demone X-Ray, che funge da proxy TCP per le chiamate API effettuate dall'SDK.

Puoi anche configurare l'SDK per caricare le regole di campionamento da un documento JSON. L'SDK può utilizzare le regole locali come backup per i casi in cui il campionamento a raggi X non è disponibile o utilizzare esclusivamente regole locali.

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
  }
}
```

Questo esempio definisce una regola personalizzata e una regola predefinita. La regola personalizzata applica una frequenza di campionamento del cinque percento senza alcun numero minimo di richieste da tracciare per i percorsi. `/api/move/` La regola predefinita tiene traccia della prima richiesta ogni secondo e del 10% delle richieste aggiuntive.

Lo svantaggio della definizione locale delle regole è che l'obiettivo fisso viene applicato da ciascuna istanza del registratore in modo indipendente, anziché essere gestito dal servizio X-Ray. Man mano che si installano più host, la tariffa fissa si moltiplica, rendendo più difficile il controllo della quantità di dati registrati.

Sì AWS Lambda, non è possibile modificare la frequenza di campionamento. Se la funzione viene chiamata da un servizio strumentato, le chiamate che hanno generato richieste campionate da quel servizio verranno registrate da Lambda. Se il tracciamento attivo è abilitato e non è presente alcuna intestazione di tracciamento, Lambda prende la decisione di campionamento.

Per configurare le regole di backup, chiedi a X-Ray SDK per Node.js di caricare le regole di campionamento da un file con. `setSamplingRules`

Example app.js - regole di campionamento da un file

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
```

Puoi anche definire le regole nel codice e passarle a `setSamplingRules` come oggetto.

Example app.js - Regole di campionamento da un oggetto

```
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);
```

Per utilizzare solo le regole locali, chiama `disableCentralizedSampling`.

```
AWSXRay.middleware.disableCentralizedSampling()
```

Registrazione dei log

Per generare i log sull'output dell'SDK, chiama `AWSXRay.setLogger(logger)`, dove `logger` è un oggetto che fornisce i metodi di log standard (`warn`, `info`, ecc.).

Per impostazione predefinita, l'SDK registrerà i messaggi di errore sulla console utilizzando i metodi standard sull'oggetto console. Il livello di registro del logger integrato può essere impostato utilizzando le variabili di `AWS_XRAY_LOG_LEVEL` ambiente `AWS_XRAY_DEBUG_MODE` o. Per un elenco di valori validi a livello di registro, vedete Variabili di ambiente.

Se desideri fornire un formato o una destinazione diversi per i log, puoi fornire all'SDK la tua implementazione dell'interfaccia del logger come mostrato di seguito. È possibile utilizzare qualsiasi oggetto che implementa questa interfaccia. Ciò significa che molte librerie di registrazione, ad esempio Winston, possono essere utilizzate e passate direttamente all'SDK.

Example app.js - logging

```
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]);
```

Chiama `setLogger` prima di eseguire altri metodi di configurazione per assicurarti che l'acquisizione dell'output di tali operazioni funzioni.

Indirizzo del demone X-Ray

Se il demone X-Ray è in ascolto su una porta o su un host diverso`127.0.0.1:2000`, è possibile configurare X-Ray SDK per Node.js per inviare i dati di traccia a un indirizzo diverso.

```
AWSXRay.setDaemonAddress('host:port');
```

È possibile specificare l'host per nome o per indirizzo. IPv4

Example app.js - Indirizzo daemon

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('daemonhost:8082');
```

Se hai configurato il daemon per ascoltare su diverse porte per TCP e UDP, puoi specificare entrambe nelle impostazioni dell'indirizzo del daemon.

Example app.js - Indirizzo del daemon su porte separate

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('tcp:daemonhost:8082 udp:daemonhost:8083');
```

È inoltre possibile impostare l'indirizzo del demone utilizzando la variabile di `AWS_XRAY_DAEMON_ADDRESS` ambiente.

Variabili di ambiente

È possibile utilizzare le variabili di ambiente per configurare l'SDK X-Ray per Node.js. L'SDK supporta le seguenti variabili.

- `AWS_XRAY_CONTEXT_MISSING`— Imposta per `RUNTIME_ERROR` generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.

Valori validi

- `RUNTIME_ERROR`— Genera un'eccezione di runtime.
- `LOG_ERROR`— Registra un errore e continua (impostazione predefinita).
- `IGNORE_ERROR`— Ignora l'errore e continua.

Gli errori relativi a segmenti o sottosegmenti mancanti possono verificarsi quando si tenta di utilizzare un client con strumenti nel codice di avvio che viene eseguito quando non è aperta alcuna richiesta o nel codice che genera un nuovo thread.

- `AWS_XRAY_DAEMON_ADDRESS`— Imposta l'host e la porta del demone X-Ray. Per impostazione predefinita, l'SDK utilizza `127.0.0.1:2000` sia i dati di traccia (UDP) che il campionamento (TCP). Utilizzate questa variabile se avete configurato il demone per l'ascolto su una porta diversa o se è in esecuzione su un host diverso.

  Formato

  - Stessa porta — *`address`*`:`*`port`*
  - Porte diverse: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_DEBUG_MODE`— Imposta su `TRUE` per configurare l'SDK per inviare i log alla console, a livello`debug`.
- `AWS_XRAY_LOG_LEVEL` — Imposta un livello di registro per il logger predefinito. I valori validi sono `debug`, `info`, `warn`, `error` e `silent`. Questo valore viene ignorato quando AWS_XRAY_DEBUG_MODE è impostato su. `TRUE`
- `AWS_XRAY_TRACING_NAME`— Imposta un nome di servizio che l'SDK utilizza per i segmenti. Sovrascrive il nome del segmento impostato sul middleware di Express.

Tracciamento delle richieste in arrivo con X-Ray SDK per Node.js

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare

a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi utilizzare l'SDK X-Ray per Node.js per tracciare le richieste HTTP in entrata che le tue applicazioni Express e Restify servono su un'istanza EC2 in Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

L'X-Ray SDK per Node.js fornisce middleware per applicazioni che utilizzano i framework Express e Restify. Quando aggiungete il middleware X-Ray all'applicazione, l'SDK X-Ray per Node.js crea un segmento per ogni richiesta campionata. Questo segmento include durata, metodo e conclusione della richiesta HTTP. Analisi ulteriori creano sottosegmenti associati a questo segmento.

> **Note**
>
> Per quanto riguarda AWS Lambda le funzioni, Lambda crea un segmento per ogni richiesta campionata. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Ogni segmento ha un nome che identifica l'applicazione nella mappa dei servizi. Il segmento può essere denominato staticamente oppure è possibile configurare l'SDK per denominarlo dinamicamente in base all'intestazione dell'host nella richiesta in entrata. La denominazione dinamica consente di raggruppare le tracce in base al nome di dominio nella richiesta e di applicare un nome predefinito se il nome non corrisponde a uno schema previsto (ad esempio, se l'intestazione dell'host è falsificata).

> **Richieste inoltrate**
>
> Se un sistema di bilanciamento del carico o un altro intermediario inoltra una richiesta all'applicazione, X-Ray prende l'IP del client dall'`X-Forwarded-For`intestazione della richiesta anziché dall'IP di origine nel pacchetto IP. L'IP del client registrato per una richiesta inoltrata può essere falsificato, quindi non dovrebbe essere considerato attendibile.

Quando viene inoltrata una richiesta, l'SDK imposta un campo aggiuntivo nel segmento per indicarlo. Se il segmento contiene il campo `x_forwarded_for` impostato su`true`, l'IP del client è stato preso dall'`X-Forwarded-For`intestazione della richiesta HTTP.

Il gestore dei messaggi crea un segmento per ogni richiesta in entrata con un blocco `http` che contiene le informazioni riportate qui di seguito:

- Metodo HTTP: GET, POST, PUT, DELETE, ecc.
- Indirizzo client: l'indirizzo IP del client che ha inviato la richiesta.
- Codice di risposta: il codice di risposta HTTP per la richiesta completata.
- Tempistica: l'ora di inizio (quando è stata ricevuta la richiesta) e l'ora di fine (quando è stata inviata la risposta).
- Agente utente: il `user-agent` codice della richiesta.
- Lunghezza del contenuto: il `content-length` risultato della risposta.

Sections

- Tracciamento delle richieste in entrata con Express
- Tracciamento delle richieste in entrata con Restify
- Configurazione di una strategia di denominazione dei segmenti

Tracciamento delle richieste in entrata con Express

Per usare il middleware Express, inizializzare il client SDK e utilizzare il middleware restituito dalla funzione `express.openSegment` prima di definire gli instradamenti.

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());
```

Dopo aver definito i percorsi, utilizzate l'output di `express.closeSegment` come mostrato per gestire eventuali errori restituiti da X-Ray SDK per Node.js.

Tracciamento delle richieste in entrata con Restify

Per usare il middleware Restify, inizializzare il client SDK client e eseguire `enable`. Passare ad esso il server Restify e il nome del 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');
});
```

Configurazione di una strategia di denominazione dei segmenti

AWS X-Ray utilizza un nome di servizio per identificare l'applicazione e distinguerla dalle altre applicazioni, database, risorse esterne APIs e AWS risorse utilizzate dall'applicazione. Quando X-Ray SDK genera segmenti per le richieste in entrata, registra il nome del servizio dell'applicazione nel campo del nome del segmento.

L'X-Ray SDK può denominare i segmenti dopo il nome host nell'intestazione della richiesta HTTP. Tuttavia, questa intestazione può essere falsificata, il che potrebbe causare nodi imprevisti nella mappa dei servizi. Per evitare che l'SDK nomini i segmenti in modo errato a causa di richieste con intestazioni host contraffatte, è necessario specificare un nome predefinito per le richieste in entrata.

Se la tua applicazione soddisfa le richieste per più domini, puoi configurare l'SDK in modo che utilizzi una strategia di denominazione dinamica che rifletta questo aspetto nei nomi dei segmenti. Una strategia di denominazione dinamica consente all'SDK di utilizzare il nome host per le richieste che corrispondono a uno schema previsto e di applicare il nome predefinito alle richieste che non lo fanno.

Ad esempio, potresti avere una singola applicazione che serve le richieste a tre sottodomini:, e. `www.example.com` `api.example.com` `static.example.com` È possibile utilizzare una strategia di denominazione dinamica con lo schema `*.example.com` per identificare i segmenti per ogni sottodominio con un nome diverso, ottenendo tre nodi di servizio sulla mappa dei servizi. Se

l'applicazione riceve richieste con un nome host che non corrisponde allo schema, sulla mappa dei servizi verrà visualizzato un quarto nodo con un nome di fallback specificato dall'utente.

Per utilizzare lo stesso nome per tutti i segmenti della richiesta, specifica il nome della tua applicazione quando inizializzi il middleware, come illustrato nelle sezioni precedenti.

> **Note**
>
> È possibile sovrascrivere il nome di servizio predefinito definito nel codice con la `AWS_XRAY_TRACING_NAME` variabile di ambiente.

Una strategia di denominazione dinamica definisce un modello al quale devono corrispondere i nomi degli host e un nome di default per l'utilizzo qualora il nome dell'host nella richiesta HTTP non corrisponda al modello. Per denominare i segmenti in modo dinamico, utilizza `AWSXRay.middleware.enableDynamicNaming`.

Example app.js - Nomi dei segmenti dinamici

Se il nome host nella richiesta corrispondente al modello `*.example.com`, utilizza il nome dell'host. In caso contrario, utilizzare `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());
```

Tracciamento delle chiamate AWS SDK con X-Ray SDK per Node.js

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del

supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate per Servizi AWS archiviare dati, scrivere in una coda o inviare notifiche, X-Ray SDK per Node.js tiene traccia delle chiamate a valle in sottosegmenti. Le risorse tracciate Servizi AWS e a cui accedi all'interno di tali servizi (ad esempio, un bucket Amazon S3 o una coda Amazon SQS) vengono visualizzate come nodi downstream sulla mappa di traccia nella console X-Ray.

Client Instrument AWS SDK creati tramite V2 o V3.AWS SDK per JavaScriptAWS SDK per JavaScript Ogni versione AWS SDK offre metodi diversi per la strumentazione dei client SDK. AWS

> **Note**
>
> Attualmente, l' AWS X-Ray SDK per Node.js restituisce meno informazioni sui segmenti durante la strumentazione dei client AWS SDK per JavaScript V3, rispetto alla strumentazione dei client V2. Ad esempio, i sottosegmenti che rappresentano le chiamate a DynamoDB non restituiranno il nome della tabella. Se hai bisogno di queste informazioni sui segmenti nelle tue tracce, prendi in considerazione l'utilizzo della versione V2. AWS SDK per JavaScript

AWS SDK per JavaScript V2

Puoi utilizzare tutti i client AWS SDK V2 inserendo l'istruzione `aws-sdk` require in una chiamata a. `AWSXRay.captureAWS`

Example app.js - Strumentazione SDK AWS

```
const AWS = AWSXRay.captureAWS(require('aws-sdk'));
```

Per strumentare i singoli client, inserisci il tuo client AWS SDK in una chiamata a. `AWSXRay.captureAWSClient` Ad esempio, per analizzare un client `AmazonDynamoDB`:

Example app.js - Strumentazione client DynamoDB

```
    const AWSXRay = require('aws-xray-sdk');
...
```

```
const ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
```

> **Warning**
>
> Non utilizzare `captureAWS` e `captureAWSClient` insieme. Ciò causerà sottosegmenti duplicati.

Se desideri utilizzarla TypeScriptcon i ECMAScriptmoduli (ESM) per caricare il JavaScript codice, usa l'esempio seguente per importare le librerie:

Example app.js - strumentazione AWS SDK

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

Per strumentare tutti i AWS client con ESM, usa il seguente codice:

Example app.js - Strumentazione AWS 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();
```

Per tutti i servizi, puoi vedere il nome dell'API richiamata nella console X-Ray. Per un sottoinsieme di servizi, l'SDK X-Ray aggiunge informazioni al segmento per fornire una maggiore granularità nella mappa dei servizi.

Ad esempio, quando si effettua una chiamata con un client DynamoDB con strumentazione, l'SDK aggiunge il nome della tabella al segmento per le chiamate destinate a una tabella. Nella console, ogni tabella appare come un nodo separato nella mappa dei servizi, con un nodo DynamoDB generico per le chiamate che non hanno come destinazione una tabella.

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
```

```
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Quando si accede alle risorse con nome, le chiamate ai seguenti servizi creano ulteriori nodi della mappa del servizio. Le chiamate che non sono hanno come obiettivo risorse specifiche creano un nodo generico per il servizio.

- Amazon DynamoDB: nome della tabella
- Amazon Simple Storage Service: nome del bucket e della chiave
- Amazon Simple Queue Service: nome della coda

AWS SDK per JavaScript V3

La AWS SDK per JavaScript V3 è modulare, quindi il codice carica solo i moduli necessari. Per questo motivo, non è possibile strumentare tutti i client AWS SDK poiché la V3 non supporta il metodo. `captureAWS`

Se desideri utilizzare TypeScript ECMAScript Modules (ESM) per caricare il JavaScript codice, puoi utilizzare il seguente esempio per importare le librerie:

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

Strumenta ogni client AWS SDK utilizzando il `AWSXRay.captureAWSv3Client` metodo. Ad esempio, per analizzare un client `AmazonDynamoDB`:

Example app.js - Strumentazione client DynamoDB che utilizza SDK per Javascript V3

```
    const AWSXRay = require('aws-xray-sdk');
```

```
    const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
...
    const ddb = AWSXRay.captureAWSv3Client(new DynamoDBClient({ region:
 "region" }));
```

Quando si utilizza AWS SDK per JavaScript V3, i metadati come il nome della tabella, il nome del bucket e della chiave o il nome della coda non vengono attualmente restituiti, pertanto la mappa di traccia non conterrà nodi discreti per ogni risorsa denominata, come accade quando si strumentano i client SDK utilizzando la V2. AWS AWS SDK per JavaScript

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento, quando si utilizza la V3 AWS SDK per JavaScript

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

Tracciamento delle chiamate ai servizi Web HTTP downstream utilizzando l'SDK X-Ray per Node.js

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare

a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate a microservizi o HTTP pubblico APIs, è possibile utilizzare il client X-Ray SDK for Node.js per strumentare tali chiamate e aggiungere l'API al grafico dei servizi come servizio downstream.

Passa il tuo `http` `https` client al `captureHTTPs` metodo X-Ray SDK for Node.js per tracciare le chiamate in uscita.

> **Note**
>
> Le chiamate che utilizzano librerie di richieste HTTP di terza parte, ad esempio Axios o Superagent, sono supportate tramite l'API `captureHTTPsGlobal()` e verranno comunque tracciate quando utilizzano il modulo `http` nativo.

Example app.js - Client HTTP

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Per abilitare il tracciamento su tutti i client HTTP, chiama `captureHTTPsGlobal` prima di caricare `http`.

Example app.js - Client HTTP (Globale)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

Quando si effettua una chiamata a un'API Web downstream, l'SDK X-Ray per Node.js registra un sottosegmento che contiene informazioni sulla richiesta e sulla risposta HTTP. X-Ray utilizza il sottosegmento per generare un segmento dedotto per l'API remota.

Example Sottosegmento per una chiamata HTTP a valle

```
{
```

```
  "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 dedotto per una chiamata HTTP a valle

```
{
  "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
}
```

Tracciamento delle query SQL con X-Ray SDK per Node.js

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Strumenta le query del database SQL inserendo il client SQL nel metodo client X-Ray SDK for Node.js corrispondente.

- 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 utilizzi un client analizzato per eseguire query SQL, X-Ray SDK per Node.js memorizza le informazioni sulla connessione e sulla query in un sottosegmento

Includere dati aggiuntivi nei sottosegmenti SQL

È possibile aggiungere informazioni aggiuntive ai sottosegmenti generati per le query SQL, purché siano mappate su un campo SQL consentito. Ad esempio, per registrare la stringa di query SQL sanificata in un sottosegmento, è possibile aggiungerla direttamente all'oggetto SQL del sottosegmento.

Example Assegna SQL al sottosegmento

```
    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;
}
```

Per un elenco completo dei campi SQL consentiti, consulta SQL Queries nella Developer Guide.AWS X-Ray

Generazione di sottosegmenti personalizzati con X-Ray SDK per Node.js

I sottosegmenti estendono il segmento di una traccia con dettagli sul lavoro svolto per soddisfare una richiesta. Ogni volta che si effettua una chiamata con un client dotato di strumentazione, l'X-Ray SDK registra le informazioni generate in un sottosegmento. È possibile creare sottosegmenti aggiuntivi per raggruppare altri sottosegmenti, misurare le prestazioni di una sezione di codice o registrare annotazioni e metadati.

Sottosegmenti Express personalizzati

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Per creare un sottosegmento personalizzato per una funzione che effettua chiamate a servizi a valle, utilizza la funzione `captureAsyncFunc`.

Example app.js - Sottosegmenti personalizzati Express

```
var AWSXRay = require('aws-xray-sdk');

app.use(AWSXRay.express.openSegment('MyApp'));

app.get('/', function (req, res) {
  var host = 'api.example.com';

  AWSXRay.captureAsyncFunc('send', function(subsegment) {
    sendRequest(host, function() {
      console.log('rendering!');
      res.render('index');
      subsegment.close();
    });
  });
});

app.use(AWSXRay.express.closeSegment());

function sendRequest(host, cb) {
  var options = {
    host: host,
    path: '/',
  };

  var callback = function(response) {
    var str = '';

    response.on('data', function (chunk) {
      str += chunk;
    });

    response.on('end', function () {
      cb();
    });
  }

  http.request(options, callback).end();
};
```

In questo esempio, l'applicazione crea un sottosegmento personalizzato denominato `send` per le chiamate alla funzione `sendRequest`. `captureAsyncFunc` trasferisce un sottosegmento che è

necessario chiudere all'interno della funzione di callback quando le chiamate asincrone invocate vengono completate.

Per le funzioni sincrone, puoi utilizzare la funzione `captureFunc` che chiude il sottosegmento automaticamente non appena il blocco della funzione si conclude.

Quando si crea un sottosegmento all'interno di un segmento o di un altro sottosegmento, X-Ray SDK per Node.js genera un ID per tale sottosegmento e registra l'ora di inizio e l'ora di fine.

Example Sottosegmento con metadati

```
"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"
    }
  },
```

Sottosegmenti Lambda personalizzati

L'SDK è configurato per creare automaticamente un segmento di facciata segnaposto quando rileva che è in esecuzione in Lambda. Per creare un sottosegmento di base, che creerà un singolo `AWS::Lambda::Function` nodo sulla mappa di tracciamento a raggi X, richiamate e riutilizzate il segmento di facciata. Se crei manualmente un nuovo segmento con un nuovo ID (mentre condividi l'ID di tracciamento, l'ID padre e la decisione di campionamento), sarai in grado di inviare un nuovo segmento.

Example app.js - sottosegmenti personalizzati manualmente

```
const segment = AWSXRay.getSegment(); //returns the facade segment
const subsegment = segment.addNewSubsegment('subseg');
...
subsegment.close();
//the segment is closed by the SDK automatically
```

Aggiungi annotazioni e metadati ai segmenti con l'SDK per X-Ray Node.js

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità di manutenzione, in cui AWS verranno limitate le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

È possibile registrare informazioni aggiuntive sulle richieste, sull'ambiente o sull'applicazione con annotazioni e metadati. Puoi aggiungere annotazioni e metadati ai segmenti creati dall' X-Ray SDK o ai sottosegmenti personalizzati che crei tu.

Le annotazioni sono coppie chiave-valore con stringhe, numeri o valori booleani. Le annotazioni sono indicizzate per essere utilizzate con le espressioni di filtro. Utilizzale per registrare i dati che desideri utilizzare per raggruppare le tracce nella console oppure per chiamare l'API `GetTraceSummaries`.

I metadati sono coppie chiave-valore che possono avere valori di qualsiasi tipo, inclusi oggetti ed elenchi, ma non sono indicizzati per essere utilizzati con le espressioni di filtro. Utilizzate i metadati per registrare dati aggiuntivi che desiderate archiviare nella traccia ma che non è necessario utilizzare con la ricerca.

Oltre ad annotazioni e metadati, sui segmenti puoi anche registrare le stringhe degli ID utente. Gli ID utente vengono memorizzati in un campo separato su segmenti e sono indicizzati per l'uso nelle ricerche.

Sections

- Registrazione delle annotazioni con l'SDK per X-Ray Node.js
- Registrazione dei metadati con l'SDK per X-Ray Node.js
- Registrazione degli ID utente con l'SDK per X-Ray Node.js

Registrazione delle annotazioni con l'SDK per X-Ray Node.js

Utilizza le annotazioni per memorizzare le informazioni su segmenti o sottosegmenti che desideri siano indicizzate per la ricerca.

Requisiti per le annotazioni

- Chiavi: la chiave per un' X-Ray annotazione può contenere fino a 500 caratteri alfanumerici. Non è possibile utilizzare spazi o simboli diversi da un punto o un punto (.)
- Valori: il valore di un' X-Ray annotazione può contenere fino a 1.000 caratteri Unicode.
- Numero di annotazioni: è possibile utilizzare fino a 50 annotazioni per traccia.

Per registrare le annotazioni

1. Ottenere un riferimento al segmento o sottosegmento corrente.

   ```
   var AWSXRay = require('aws-xray-sdk');
   ...
   var document = AWSXRay.getSegment();
   ```

2. Chiamare `addAnnotation` con una chiave di tipo Stringa e un valore booleano, numerico o di tipo Stringa.

   ```
   document.addAnnotation("mykey", "my value");
   ```

   L'esempio seguente mostra come effettuare una chiamata `addAnnotation` con una chiave String che include un punto e un valore booleano, numerico o stringa.

   ```
   document.addAnnotation("testkey.test", "my value");
   ```

L'SDK memorizza le annotazioni come coppie chiave-valore in un oggetto `annotations` all'interno del documento di segmento. Se chiami `addAnnotation` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Per trovare tracciamenti con annotazioni contenenti valori specifici, utilizza la parola chiave `annotation[`*`key`*`]` in un'espressione filtro.

Example app.js - Annotazioni

```
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) {
...
```

Registrazione dei metadati con l'SDK per X-Ray Node.js

Utilizza i metadati per memorizzare le informazioni su segmenti o sottosegmenti che non è necessario che siano indicizzate per la ricerca. I valori dei metadati possono essere stringhe, numeri, valori booleani o qualsiasi altro oggetto che può essere serializzato in un oggetto o in un vettore JSON.

Per registrare i metadati

1. Ottenere un riferimento al segmento o sottosegmento corrente.

   ```
   var AWSXRay = require('aws-xray-sdk');
   ...
   var document = AWSXRay.getSegment();
   ```

2. Chiamare `addMetadata` con una chiave di tipo Stringa, un valore booleano, numerico, di tipo Stringa o di tipo oggetto e un namespace di tipo stringa.

```
document.addMetadata("my key", "my value", "my namespace");
```

or

Invocare `addMetadata` semplicemente con una chiave e un valore.

```
document.addMetadata("my key", "my value");
```

Se non specifichi un namespace, l'SDK utilizza `default`. Se chiami `addMetadata` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Registrazione degli ID utente con l'SDK per X-Ray Node.js

Memorizza gli ID utente sui segmenti di richiesta per identificare l'utente che ha inviato la richiesta. Questa operazione non è compatibile con AWS Lambda le funzioni perché i segmenti negli ambienti Lambda sono immutabili. La chiamata `setUser` può essere applicata solo ai segmenti, non ai sottosegmenti.

Per registrare gli ID degli utenti

1. Ottenere un riferimento al segmento o sottosegmento corrente.

```
var AWSXRay = require('aws-xray-sdk');
...
var document = AWSXRay.getSegment();
```

2. Chiamare `setUser()` con una stringa che rappresenta l'ID dell'utente che ha inviato la richiesta.

```
var user = 'john123';

AWSXRay.getSegment().setUser(user);
```

È possibile chiamare `setUser` per registrare l'ID utente non appena l'applicazione Express inizia l'elaborazione di una richiesta. Se usi il segmento per impostare l'ID utente, puoi concatenare le chiamate in un'unica riga.

Example app.js - ID utente

```
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) {
...
```

Per trovare tracciamenti associati ad un ID utente, utilizza la parola chiave `user` in un'espressione filtro.

Uso di Python

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Esistono due modi per strumentare un'applicazione Python per inviare tracce a X-Ray:

- AWS Distro for OpenTelemetry Python: AWS una distribuzione che fornisce un set di librerie open source per l'invio di metriche e tracce correlate a AWS più soluzioni di monitoraggio, tra cui Amazon e OpenSearch Amazon Service CloudWatch AWS X-Ray, tramite AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK per Python — Un insieme di librerie per generare e inviare tracce a X-Ray tramite il demone X-Ray.

Per ulteriori informazioni, consulta Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs.

AWS Distro per Python OpenTelemetry

Con AWS Distro for OpenTelemetry (ADOT) Python, puoi strumentare le tue applicazioni una sola volta e inviare metriche e tracce correlate a più AWS soluzioni di monitoraggio tra cui Amazon e Amazon Service. CloudWatch AWS X-Ray OpenSearch L'utilizzo di X-Ray con ADOT richiede due componenti: un OpenTelemetry SDK abilitato per l'uso con X-Ray e AWS Distro for OpenTelemetry Collector abilitato per l'uso con X-Ray. ADOT Python include il supporto per la strumentazione automatica, che consente all'applicazione di inviare tracce senza modifiche al codice.

Per iniziare, consulta la documentazione di AWS Distro for OpenTelemetry Python.

Per ulteriori informazioni sull'utilizzo di AWS Distro for OpenTelemetry with AWS X-Ray e altro Servizi AWS, consulta AWS Distro for OpenTelemetry o Distro for Documentation.AWS OpenTelemetry

Per ulteriori informazioni sul supporto e l'utilizzo della lingua, vedere AWS Observability on. GitHub

AWS X-Ray SDK for Python

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

L' X-Ray SDK per Python è una libreria per applicazioni web Python che fornisce classi e metodi per generare e inviare dati di traccia al demone. X-Ray I dati di traccia includono informazioni sulle richieste HTTP in entrata servite dall'applicazione e sulle chiamate che l'applicazione effettua ai servizi downstream utilizzando l' AWS SDK, i client HTTP o un connettore di database SQL. Puoi anche possibile creare manualmente dei segmenti e aggiungere informazioni di debug in annotazioni e metadati.

Puoi scaricare l'SDK tramite `pip`.

```
$ pip install aws-xray-sdk
```

> **Note**
>
> L' X-Ray SDK per Python è un progetto open source. Puoi seguire il progetto e inviare problemi e richieste pull su GitHub: github. com/aws/aws-xray-sdk-python

Se utilizzi Django o Flask, per tracciare le richieste in entrata inizia aggiungendo il middleware dell'SDK alla tua applicazione. Il middleware crea un segmento per ogni richiesta tracciata e completa il segmento quando viene inviata la risposta. Fino a che il segmento è aperto, puoi usare i metodi del client dell'SDK per aggiungere informazioni al segmento e creare sottosegmenti per tracciare le chiamate a valle. L'SDK, inoltre, registra automaticamente le eccezioni sollevate dall'applicazione per il tempo durante il quale il segmento è aperto. Per altre applicazioni, puoi creare i segmenti manualmente.

Per le funzioni Lambda richiamate da un'applicazione o un servizio con strumenti, Lambda legge l'intestazione di tracciamento e traccia automaticamente le richieste campionate. ??? Per altre funzioni, puoi configurare Lambda per campionare e tracciare le richieste in arrivo. In entrambi i casi, Lambda crea il segmento e lo fornisce all'SDK. X-Ray

> **Note**
>
> Su Lambda, l' X-Ray SDK è opzionale. Se non lo usi nella tua funzione, la mappa del servizio includerà comunque un nodo per il servizio Lambda e uno per ogni funzione Lambda. Aggiungendo l'SDK, puoi utilizzare il codice della funzione per aggiungere sottosegmenti al segmento di funzione registrato da Lambda. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Vedi Worker per un esempio di funzione Python strumentata in Lambda.

Quindi, usa l' X-Ray SDK per Python per strumentare le chiamate a valle applicando patch alle librerie della tua applicazione. L'SDK supporta le seguenti librerie.

Librerie supportate

- `botocore`, `boto3` — Client per strumenti. AWS SDK per Python (Boto)
- `pynamodb`— La versione di Instrument PynamoDB del client Amazon DynamoDB.
- `aiobotocore`, `aioboto3` — Instrument asyncio: versioni integrate dell'SDK per client Python (Boto3).
- `requests`, — Client HTTP di alto livello per Instrument. `aiohttp`
- `httplib`, `http.client` — Client HTTP di basso livello dello strumento e librerie di livello superiore che li utilizzano.
- `sqlite3`— Client SQLite di Instrument.
- `mysql-connector-python`— Client MySQL di Instrument.
- `pg8000`— Interfaccia Pure-Python PostgreSQL dello strumento.
- `psycopg2`— Adattatore per database PostgreSQL Instrument.
- `pymongo`— Client MongoDB per strumenti.
- `pymysql`— Client basati su PyMy SQL di Instrument per MySQL e MariaDB.

Ogni volta che l'applicazione effettua chiamate a AWS, a un database SQL o ad altri servizi HTTP, l'SDK registra le informazioni sulla chiamata in un sottosegmento. Servizi AWS e le risorse a cui accedi all'interno dei servizi vengono visualizzate come nodi a valle sulla mappa di traccia per aiutarti a identificare errori e problemi di limitazione sulle singole connessioni.

Dopo aver iniziato a utilizzare l'SDK, personalizzane il comportamento configurando il registratore e il middleware. Puoi aggiungere dei plugin per memorizzare i dati sulle risorse di elaborazione sulle quali è eseguita la tua applicazione, personalizzare il comportamento di campionamento definendo regole di campionatura e impostare il livello di log per visualizzare più o meno informazioni generate dall'SDK nei log dell'applicazione.

Registra ulteriori informazioni sulle richieste e sull'attività svolta dalla tua applicazione in annotazioni e metadati. Le annotazioni sono semplici coppie chiave-valore indicizzati per l'uso con espressioni filtro, in modo da poter cercare tracce che contengono dati specifici. Le immissioni di metadati sono meno restrittive e possono registrare interi oggetti e array, qualsiasi cosa che possa essere serializzata in JSON.

> **Annotazioni e metadata**
>
> Le annotazioni e i metadati sono testo arbitrario che aggiungi ai segmenti con l'SDK. X-Ray Le annotazioni sono indicizzate per essere utilizzate con le espressioni di filtro. I metadati non sono indicizzati, ma possono essere visualizzati nel segmento raw con la console o l'API. X-Ray Chiunque conceda l'accesso in lettura X-Ray può visualizzare questi dati.

Quando disponi di una notevole quantità di client analizzati nel tuo codice, un singolo segmento per la richiesta può contenere un numero elevato di sottosegmenti, uno per ciascuna delle chiamate effettuate con un client analizzato. Puoi organizzare e raggruppare i sottosegmenti inglobando le chiamate del client sottosegmenti personalizzati. Puoi creare un sottosegmento personalizzato per un'intera funzione o per qualsiasi parte del codice. Puoi quindi memorizzare metadati e annotazioni nel sottosegmento invece di scrivere tutto all'interno del segmento padre.

Per la documentazione di riferimento per le classi e i metodi dell'SDK, consulta l'AWS X-Ray SDK for Python API Reference.

Requisiti

L' X-Ray SDK per Python supporta le seguenti versioni del linguaggio e della libreria.

- Python: 2.7, 3.4 e versioni successive
- Django — 1.10 e versioni successive
- Flask — 0.10 e versioni successive
- aiohttp — 2.3.0 e versioni successive
- AWS SDK per Python (Boto)— 1.4.0 e versioni successive
- botocore — 1.5.0 e versioni successive
- enum — 0.4.7 e versioni successive, per le versioni di Python 3.4.0 e precedenti
- jsonpickle — 1.0.0 e versioni successive
- setuptools — 40.6.3 e versioni successive
- wrapt — 1.11.0 e versioni successive

Gestione delle dipendenze

L' X-Ray SDK per Python è disponibile da. `pip`

- Pacchetto: `aws-xray-sdk`

Aggiungi l'SDK come dipendenza nel file `requirements.txt`.

Example requirements.txt

```
aws-xray-sdk==2.4.2
boto3==1.4.4
botocore==1.5.55
Django==1.11.3
```

Se si utilizza Elastic Beanstalk per distribuire l'applicazione, Elastic Beanstalk installa automaticamente tutti i pacchetti. `requirements.txt`

Configurazione dell' X-Ray SDK per Python

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS verranno limitate le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni

sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support
Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

L' X-Ray SDK per Python ha una classe `xray_recorder` denominata che fornisce il registratore globale. Puoi configurare il registratore globale per personalizzare il middleware che crea i segmenti relativi alle chiamate HTTP in entrata.

Sections

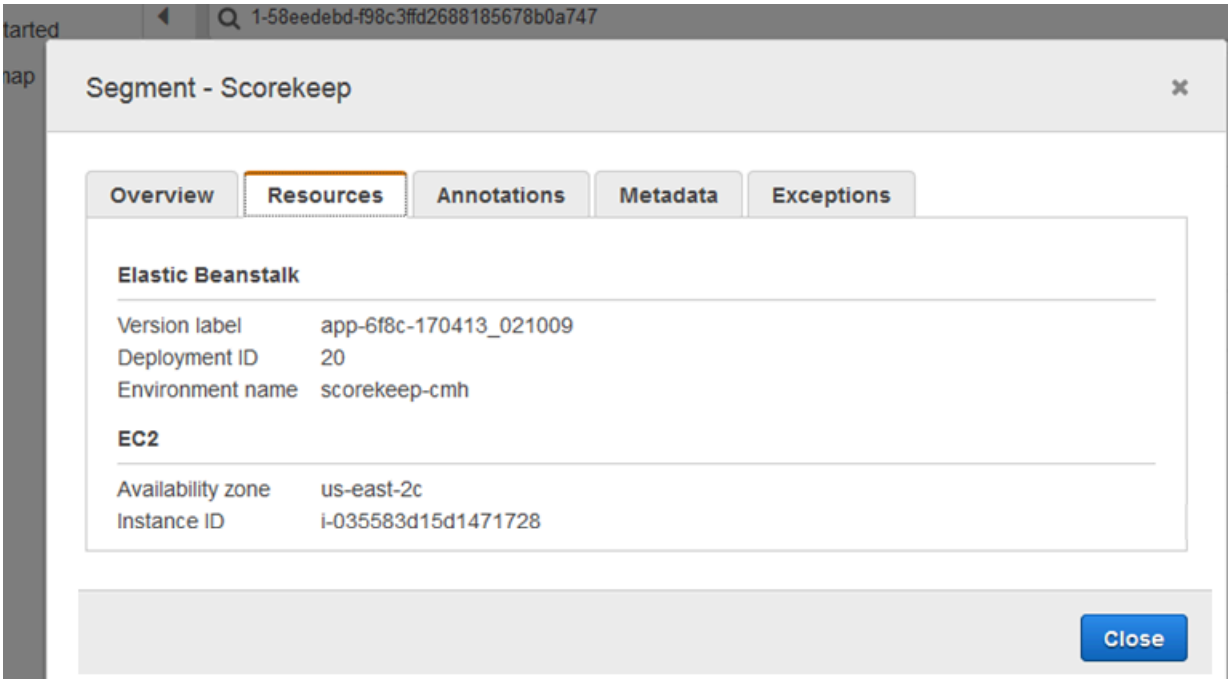
- Plugin di servizio
- Regole di campionamento
- Registrazione dei log
- Configurazione del registratore nel codice
- Configurazione del registratore con Django
- Variabili di ambiente

Plugin di servizio

`plugins`Utilizzatelo per registrare informazioni sul servizio che ospita l'applicazione.

Plugin

- Amazon EC2: `EC2Plugin` aggiunge l'ID dell'istanza, la zona di disponibilità e il gruppo di CloudWatch log.
- Elastic `ElasticBeanstalkPlugin` Beanstalk: aggiunge il nome dell'ambiente, l'etichetta della versione e l'ID di distribuzione.
- Amazon ECS: `ECSPlugin` aggiunge l'ID del contenitore.



Per usare un plugin, chiama `configure` sul tuo `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**
>
> Poiché `plugins` vengono passati come tupla, assicurati di includere un `,` trailing quando specifichi un singolo plugin. Ad esempio, `plugins = ('EC2Plugin',)`

Per configurare il registratore puoi anche utilizzare le variabili di ambiente, che hanno la precedenza sui valori impostati nel codice.

Configura i plugin prima applicare le patch alle librerie per memorizzare le chiamate a valle.

L'SDK utilizza anche le impostazioni del plug-in per impostare il `origin` campo sul segmento. Indica il tipo di AWS risorsa che esegue l'applicazione. Quando utilizzi più plugin, l'SDK utilizza il seguente ordine di risoluzione per determinare l'origine: ElasticBeanstalk > EKS > ECS > EC2.

Regole di campionamento

L'SDK utilizza le regole di campionamento definite nella X-Ray console per determinare quali richieste registrare. La regola predefinita tiene traccia della prima richiesta ogni secondo e del 5% di tutte le richieste aggiuntive su tutti i servizi a cui vengono inviate le tracce. X-Ray Crea regole aggiuntive nella X-Ray console per personalizzare la quantità di dati registrati per ciascuna delle tue applicazioni.

L'SDK applica le regole personalizzate nell'ordine in cui sono definite. Se una richiesta corrisponde a più regole personalizzate, l'SDK applica solo la prima regola.

> **Note**
>
> Se l'SDK non riesce X-Ray a ottenere le regole di campionamento, torna a una regola locale predefinita che prevede la prima richiesta ogni secondo e il 5% di eventuali richieste aggiuntive per host. Ciò può verificarsi se l'host non dispone dell'autorizzazione per chiamare le API di campionamento o non può connettersi al X-Ray demone, che funge da proxy TCP per le chiamate API effettuate dall'SDK.

Puoi anche configurare l'SDK per caricare le regole di campionamento da un documento JSON. L'SDK può utilizzare le regole locali come backup per i casi in cui il X-Ray campionamento non è disponibile o utilizzare esclusivamente regole locali.

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
  }
}
```

Questo esempio definisce una regola personalizzata e una regola predefinita. La regola personalizzata applica una frequenza di campionamento del cinque percento senza alcun numero minimo di richieste da tracciare per i percorsi. `/api/move/` La regola predefinita tiene traccia della prima richiesta ogni secondo e del 10% delle richieste aggiuntive.

Lo svantaggio della definizione locale delle regole è che l'obiettivo fisso viene applicato da ciascuna istanza del registratore in modo indipendente, anziché essere gestito dal X-Ray servizio. Man mano che si implementano più host, la tariffa fissa si moltiplica, rendendo più difficile il controllo della quantità di dati registrati.

Sì AWS Lambda, non è possibile modificare la frequenza di campionamento. Se la funzione viene chiamata da un servizio strumentato, le chiamate che hanno generato richieste campionate da quel servizio verranno registrate da Lambda. Se il tracciamento attivo è abilitato e non è presente alcuna intestazione di tracciamento, Lambda prende la decisione di campionamento.

Per configurare le regole di campionamento di backup, chiamate`xray_recorder.configure`, come illustrato nell'esempio seguente, where *`rules`* è un dizionario di regole o il percorso assoluto di un file JSON contenente le regole di campionamento.

```
xray_recorder.configure(sampling_rules=rules)
```

Per utilizzare solo regole locali, configura un registratore con una `LocalSampler`.

```
from aws_xray_sdk.core.sampling.local.sampler import LocalSampler
xray_recorder.configure(sampler=LocalSampler())
```

Puoi anche configurare il registratore globale per disabilitare il campionamento e analizzare tutte le richieste in entrata.

Example main.py — Disabilita il campionamento

```
xray_recorder.configure(sampling=False)
```

Registrazione dei log

L'SDK utilizza il `logging` modulo integrato di Python con un livello di registrazione predefinitoWARNING. Ottieni un riferimento al logger per la classe `aws_xray_sdk` e chiama `setLevel` per configurare il livello di log differenti per la libreria e il resto dell'applicazione.

Example app.py — Registrazione

```
logging.basicConfig(level='WARNING')
logging.getLogger('aws_xray_sdk').setLevel(logging.ERROR)
```

Utilizza i log di debug per identificare i problemi, come ad esempio dei sottosegmenti non chiusi, quando generi dei sottosegmenti manualmente.

Configurazione del registratore nel codice

Impostazioni aggiuntive sono disponibili tramite il metodo `configure` su `xray_recorder`.

- `context_missing`— Imposta per LOG_ERROR evitare di generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.
- `daemon_address`— Imposta l'host e la porta del daemon listener. X-Ray
- `service`— Imposta un nome di servizio che l'SDK utilizza per i segmenti.
- `plugins`— Registra informazioni sulle risorse dell'applicazione. AWS
- `sampling`— Impostato per `False` disabilitare il campionamento.
- `sampling_rules`— Imposta il percorso del file JSON contenente le regole di campionamento.

Example main.py — Disabilita le eccezioni mancanti nel contesto

```
from aws_xray_sdk.core import xray_recorder

xray_recorder.configure(context_missing='LOG_ERROR')
```

Configurazione del registratore con Django

Se utilizzi il framework Django, puoi utilizzare il file `settings.py` di Django per configurare le opzioni del registratore globale.

- `AUTO_INSTRUMENT`(Solo Django) — Registra i sottosegmenti per le operazioni di rendering di database e modelli integrate.
- `AWS_XRAY_CONTEXT_MISSING`— Imposta per `LOG_ERROR` evitare di generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.
- `AWS_XRAY_DAEMON_ADDRESS`— Imposta l'host e la porta del daemon listener. X-Ray
- `AWS_XRAY_TRACING_NAME`— Imposta un nome di servizio che l'SDK utilizza per i segmenti.
- `PLUGINS`— Registra informazioni sulle risorse dell'applicazione. AWS
- `SAMPLING`— Impostato per `False` disabilitare il campionamento.
- `SAMPLING_RULES`— Imposta il percorso del file JSON contenente le regole di campionamento.

Per abilitare la configurazione del registratore in `settings.py`, aggiungi il middleware Django all'elenco delle applicazioni.

Example settings.py — App installate

```
INSTALLED_APPS = [
    ...
    'django.contrib.sessions',
    'aws_xray_sdk.ext.django',
]
```

Configura le impostazioni disponibili in un dizionario denominato `XRAY_RECORDER`.

Example settings.py — App installate

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_CONTEXT_MISSING': 'LOG_ERROR',
    'AWS_XRAY_DAEMON_ADDRESS': '127.0.0.1:5000',
    'AWS_XRAY_TRACING_NAME': 'My application',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin', 'ECSPlugin'),
    'SAMPLING': False,
}
```

Variabili di ambiente

Puoi usare le variabili di ambiente per configurare l' X-Ray SDK per Python. L'SDK supporta le seguenti variabili:

- `AWS_XRAY_TRACING_NAME`— Imposta un nome di servizio che l'SDK utilizza per i segmenti. Sostituisce il nome del servizio impostato a livello di programmazione.
- `AWS_XRAY_SDK_ENABLED`— Se impostato su`false`, disabilita l'SDK. Per impostazione predefinita, l'SDK è abilitato solo se la variabile di ambiente è impostata su false.
  - Quando l'SDK è disabilitato, il registratore globale genera automaticamente segmenti o sottosegmenti fittizi che non vengono inviati al daemon e l'applicazione automatica di patch è disattivata. I middleware vengono scritti come wrapper per il registratore globale. Anche tutti i segmenti e sottosegmenti generati tramite il middleware diventano segmenti e sottosegmenti fittizi.
  - Imposta il valore di `AWS_XRAY_SDK_ENABLED` tramite la variabile di ambiente o l'interazione diretta con l'oggetto `global_sdk_config` dalla libreria `aws_xray_sdk`. Le impostazioni con la variabile di ambiente sostituiscono le interazioni.
- `AWS_XRAY_DAEMON_ADDRESS`— Imposta l'host e la porta del X-Ray daemon listener. Per impostazione predefinita, l'SDK utilizza sia i dati `127.0.0.1:2000` di traccia (UDP) che il campionamento (TCP). Utilizzate questa variabile se avete configurato il demone per l'ascolto su una porta diversa o se è in esecuzione su un host diverso.

  Formato

  - Stessa porta — *`address`*`:`*`port`*
  - Porte diverse: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_CONTEXT_MISSING`— Imposta `RUNTIME_ERROR` per generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.

  Valori validi

  - `RUNTIME_ERROR`— Genera un'eccezione di runtime.
  - `LOG_ERROR`— Registra un errore e continua (impostazione predefinita).
  - `IGNORE_ERROR`— Ignora l'errore e continua.

  Gli errori relativi a segmenti o sottosegmenti mancanti possono verificarsi quando si tenta di utilizzare un client con strumenti nel codice di avvio che viene eseguito quando non è aperta alcuna richiesta o nel codice che genera un nuovo thread.

Le variabili di ambiente sostituiscono i valori impostati nel codice.

Tracciamento delle richieste in arrivo con il middleware X-Ray SDK per Python

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando aggiungi il middleware all'applicazione e configuri il nome di un segmento, X-Ray SDK for Python crea un segmento per ogni richiesta campionata. Questo segmento include durata, metodo e conclusione della richiesta HTTP. Analisi ulteriori creano sottosegmenti associati a questo segmento.

L'X-Ray SDK per Python supporta il seguente middleware per strumentare le richieste HTTP in arrivo:

- Django
- Flask
- Bottle

> **Note**
>
> Per quanto riguarda AWS Lambda le funzioni, Lambda crea un segmento per ogni richiesta campionata. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Vedi un Worker esempio di funzione Python strumentata in Lambda.

Per gli script o le applicazioni Python su altri framework, puoi creare i segmenti manualmente.

Ogni segmento ha un nome che identifica l'applicazione nella mappa dei servizi. Il segmento può essere denominato staticamente oppure è possibile configurare l'SDK per denominarlo dinamicamente in base all'intestazione dell'host nella richiesta in entrata. La denominazione dinamica

consente di raggruppare le tracce in base al nome di dominio nella richiesta e di applicare un nome predefinito se il nome non corrisponde a uno schema previsto (ad esempio, se l'intestazione dell'host è falsificata).

> **Richieste inoltrate**
>
> Se un sistema di bilanciamento del carico o un altro intermediario inoltra una richiesta all'applicazione, X-Ray prende l'IP del client dall'`X-Forwarded-For`intestazione della richiesta anziché dall'IP di origine nel pacchetto IP. L'IP del client registrato per una richiesta inoltrata può essere falsificato, quindi non dovrebbe essere considerato attendibile.

Quando viene inoltrata una richiesta, l'SDK imposta un campo aggiuntivo nel segmento per indicarlo. Se il segmento contiene il campo `x_forwarded_for` impostato su`true`, l'IP del client è stato preso dall'`X-Forwarded-For`intestazione della richiesta HTTP.

Il middleware crea un segmento per ogni richiesta in entrata con un blocco `http` che contiene le informazioni riportate qui di seguito:

- Metodo HTTP: GET, POST, PUT, DELETE, ecc.
- Indirizzo client: l'indirizzo IP del client che ha inviato la richiesta.
- Codice di risposta: il codice di risposta HTTP per la richiesta completata.
- Tempistica: l'ora di inizio (quando è stata ricevuta la richiesta) e l'ora di fine (quando è stata inviata la risposta).
- Agente utente: il `user-agent` codice della richiesta.
- Lunghezza del contenuto: il `content-length` risultato della risposta.

Sections

- Aggiunta del middleware all'applicazione (Django)
- Aggiunta del middleware all'applicazione (flask)
- Aggiunta del middleware all'applicazione (Bottle)
- Analisi manuale del codice Python
- Configurazione di una strategia di denominazione dei segmenti

Aggiunta del middleware all'applicazione (Django)

Aggiungi il middleware all'elenco `MIDDLEWARE` nel tuo file `settings.py`. Il middleware X-Ray dovrebbe essere presente nella prima riga del tuo file `settings.py` per assicurarti che le richieste che non ricadano sotto il controllo di altri middleware quando vengono registrate.

Example settings.py - X-Ray SDK per middleware 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'
]
```

Aggiungi l'app X-Ray SDK Django all'`INSTALLED_APP`Selenco del tuo file. `settings.py` Ciò consentirà di configurare il registratore X-Ray durante l'avvio dell'app.

Example settings.py - X-Ray SDK per l'app Python Django

```
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',
]
```

Configurazione di un nome di segmento nel tuo file `settings.py`.

Example settings.py — Nome del segmento

```
XRAY_RECORDER = {
    'AWS_XRAY_TRACING_NAME': 'My application',
    'PLUGINS': ('EC2Plugin',),
}
```

Questo dice al registratore X-Ray di tracciare le richieste servite dall'applicazione Django con la frequenza di campionamento predefinita. Puoi configurare il file delle impostazioni del registratore Django affinché applichi le regole di campionatura personalizzate o modificare altre impostazioni.

> **Note**
>
> Poiché `plugins` vengono passate come tupla, assicuratevi di includere un finale quando specificate un singolo plugin. `,` Ad esempio, `plugins = ('EC2Plugin',)`

Aggiunta del middleware all'applicazione (flask)

Per analizzare la tua applicazione, prima di tutto configura un nome di segmento in `xray_recorder`. Quindi, utilizza la funzione `XRayMiddleware` per applicare una patch al codice dell'applicazione Flask.

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)
```

Questo indica al registratore X-Ray di tracciare le richieste servite dall'applicazione Flask con la frequenza di campionamento predefinita. Puoi configurare il registratore all'interno del codice affinché applichi le regole di campionatura personalizzate o modificare altre impostazioni.

Aggiunta del middleware all'applicazione (Bottle)

Per analizzare l'applicazione Bottle, prima di tutto configura un nome di segmento in `xray_recorder`. Quindi, utilizza la funzione `XRayMiddleware` per applicare una patch al codice dell'applicazione Bottle.

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))
```

Questo indica al registratore X-Ray di tracciare le richieste servite dall'applicazione Bottle con la frequenza di campionamento predefinita. Puoi configurare il registratore all'interno del codice affinché applichi le regole di campionatura personalizzate o modificare altre impostazioni.

Analisi manuale del codice Python

Se non utilizzi Django o Flask, puoi creare i segmenti manualmente. È possibile creare un segmento per ogni richiesta in arrivo o creare segmenti attorno ai client HTTP o AWS SDK con patch per fornire al registratore il contesto in cui aggiungere sottosegmenti.

Example main.py — Strumentazione manuale

```
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()
```

Configurazione di una strategia di denominazione dei segmenti

AWS X-Ray utilizza un nome di servizio per identificare l'applicazione e distinguerla dalle altre applicazioni, database, AWS risorse esterne APIs e utilizzate dall'applicazione. Quando X-Ray SDK genera segmenti per le richieste in entrata, registra il nome del servizio dell'applicazione nel campo del nome del segmento.

L'X-Ray SDK può denominare i segmenti dopo il nome host nell'intestazione della richiesta HTTP. Tuttavia, questa intestazione può essere falsificata, il che potrebbe causare nodi imprevisti nella

mappa dei servizi. Per evitare che l'SDK nomini i segmenti in modo errato a causa di richieste con intestazioni host contraffatte, è necessario specificare un nome predefinito per le richieste in entrata.

Se la tua applicazione soddisfa le richieste per più domini, puoi configurare l'SDK in modo che utilizzi una strategia di denominazione dinamica che rifletta questo aspetto nei nomi dei segmenti. Una strategia di denominazione dinamica consente all'SDK di utilizzare il nome host per le richieste che corrispondono a uno schema previsto e di applicare il nome predefinito alle richieste che non lo fanno.

Ad esempio, potresti avere una singola applicazione che serve le richieste a tre sottodomini:, e. `www.example.com` `api.example.com` `static.example.com` È possibile utilizzare una strategia di denominazione dinamica con lo schema `*.example.com` per identificare i segmenti per ogni sottodominio con un nome diverso, ottenendo tre nodi di servizio sulla mappa dei servizi. Se l'applicazione riceve richieste con un nome host che non corrisponde allo schema, sulla mappa dei servizi verrà visualizzato un quarto nodo con un nome di fallback specificato dall'utente.

Per utilizzare lo stesso nome per tutti i segmenti della richiesta, specifica il nome della tua applicazione quando configuri il registratore, come illustrato nelle sezioni precedenti.

Una strategia di denominazione dinamica definisce un modello al quale devono corrispondere i nomi degli host e un nome di default per l'utilizzo qualora il nome dell'host nella richiesta HTTP non corrisponda al modello. Per denominare in modo dinamico i segmenti in Django, aggiungere l'impostazione `DYNAMIC_NAMING` al tuo file settings.py.

Example settings.py — Denominazione dinamica

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_TRACING_NAME': 'My application',
    'DYNAMIC_NAMING': '*.example.com',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin')
}
```

Puoi utilizzare '*' nel modello per una corrispondenza con qualsiasi stringa o '?' per una corrispondenza con qualsiasi carattere singolo. Per Flask, configura il registratore nel codice.

Example main.py — Nome del segmento

```
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='My application')
```

```
xray_recorder.configure(dynamic_naming='*.example.com')
```

> **Note**
>
> È possibile sovrascrivere il nome di servizio predefinito definito nel codice con la variabile di `AWS_XRAY_TRACING_NAME` ambiente.

Applicare le patch alle librerie per analizzare le chiamate a valle

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità manutenzione, che AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica dell'assistenza, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry

Per strumentare le chiamate downstream, usate l' X-Ray SDK for Python per patchare le librerie utilizzate dall'applicazione. L' X-Ray SDK per Python può applicare patch alle seguenti librerie.

Librerie supportate

- `botocore`, `boto3` — Client per strumenti AWS SDK per Python (Boto) .
- `pynamodb`— La versione di Instrument PynamoDB del client Amazon DynamoDB.
- `aiobotocore`, `aioboto3` — Instrument asyncio: versioni integrate dell'SDK per client Python (Boto3).
- `requests`, — Client HTTP di alto livello per Instrument. `aiohttp`
- `httplib`, `http.client` — Client HTTP di basso livello dello strumento e librerie di livello superiore che li utilizzano.
- `sqlite3`— Client SQLite di Instrument.
- `mysql-connector-python`— Client MySQL di Instrument.
- `pg8000`— Interfaccia Pure-Python PostgreSQL dello strumento.
- `psycopg2`— Adattatore per database PostgreSQL Instrument.

- `pymongo`— Client MongoDB per strumenti.
- `pymysql`— Client basati su PyMy SQL di Instrument per MySQL e MariaDB.

Quando si utilizza una libreria patchata, l' X-Ray SDK per Python crea un sottosegmento per la chiamata e registra le informazioni relative alla richiesta e alla risposta. Un segmento devono essere disponibili affinché l'SDK possa creare il sottosegmento, sia dal middleware dell'SDK che da AWS Lambda.

> **Note**
>
> Se utilizzi SQLAlchemy ORM, puoi analizzare le tue query SQL importando la versione dell'SDK delle classi della sessione e delle query di SQLAlchemy. Consulta Utilizzare SQLAlchemy ORM per le relative istruzioni.

Per installare le patch disponibili per tutte le librerie, utilizza la funzione `patch_all` in `aws_xray_sdk.core`. Alcune librerie, ad esempio `httplib` e `urllib`, potrebbero dover abilitare l'applicazione di patch doppie chiamando `patch_all(double_patch=True)`.

Example main.py — Correggi tutte le librerie supportate

```
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()
```

Per applicare patch a una singola libreria, chiamare `patch` con una tupla del nome della libreria. Per raggiungere questo risultato, è necessario fornire un singolo elenco di elementi.

Example main.py — Librerie specifiche per le patch

```
import boto3
import botocore
import requests
```

```
import mysql-connector-python

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch

libraries = (['botocore'])
patch(libraries)
```

> **Note**
>
> In alcuni casi, la chiave utilizzata per installare le patch di una libreria non corrisponde al nome della libreria. Alcune chiavi servono come alias per una o più librerie.
>
> Alias librerie
>
> - `httplib`: `httplib` e `http.client`
> - `mysql` – `mysql-connector-python`

Tracciamento del contesto per le attività asincrone

Per le librerie `asyncio` integrate o per creare sottosegmenti per funzioni asincrone, devi anche configurare l' X-Ray SDK per Python con un contesto asincrono. Importa la `AsyncContext` classe e passane un'istanza al registratore. X-Ray

> **Note**
>
> Le librerie di supporto del framework Web, come AIOHTTP, non vengono gestite tramite il modulo `aws_xray_sdk.core.patcher`. Non sono presenti nel catalogo `patcher` delle librerie supportate.

Example main.py — Patch aioboto3

```
import asyncio
import aioboto3
import requests

from aws_xray_sdk.core.async_context import AsyncContext
from aws_xray_sdk.core import xray_recorder
```

```
xray_recorder.configure(service='my_service', context=AsyncContext())
from aws_xray_sdk.core import patch

libraries = (['aioboto3'])
patch(libraries)
```

Tracciamento delle chiamate AWS SDK con X-Ray SDK per Python

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate per Servizi AWS archiviare dati, scrivere in una coda o inviare notifiche, X-Ray SDK for Python tiene traccia delle chiamate a valle in sottosegmenti. Le risorse tracciate Servizi AWS e a cui accedi all'interno di tali servizi (ad esempio, un bucket Amazon S3 o una coda Amazon SQS) vengono visualizzate come nodi downstream sulla mappa di traccia nella console X-Ray.

L'X-Ray SDK per Python strumenta automaticamente tutti i client AWS SDK quando applichi una patch alla libreria. `botocore` Non puoi analizzare singoli client.

Per tutti i servizi, puoi vedere il nome dell'API richiamata nella console X-Ray. Per un sottoinsieme di servizi, l'SDK X-Ray aggiunge informazioni al segmento per fornire una maggiore granularità nella mappa dei servizi.

Ad esempio, quando si effettua una chiamata con un client DynamoDB con strumentazione, l'SDK aggiunge il nome della tabella al segmento per le chiamate destinate a una tabella. Nella console, ogni tabella appare come un nodo separato nella mappa dei servizi, con un nodo DynamoDB generico per le chiamate che non hanno come destinazione una tabella.

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento

```
{
```

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

Quando si accede alle risorse con nome, le chiamate ai seguenti servizi creano ulteriori nodi della mappa del servizio. Le chiamate che non sono hanno come obiettivo risorse specifiche creano un nodo generico per il servizio.

- Amazon DynamoDB: nome della tabella
- Amazon Simple Storage Service: nome del bucket e della chiave
- Amazon Simple Queue Service: nome della coda

Tracciamento delle chiamate ai servizi Web HTTP downstream utilizzando l'SDK X-Ray per Python

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate a microservizi o a HTTP pubblico APIs, è possibile utilizzare l'X-Ray SDK for Python per strumentare tali chiamate e aggiungere l'API al service graph come servizio downstream.

Per analizzare i client HTTP, applica le patch alla libreria che utilizzi per effettuare le chiamate in uscita. Se utilizzi `requests` o il client HTTP integrato in Python, questo è tutto ciò che devi fare. Per `aiohttp`, inoltre, configura il registratore con un contesto async.

Se utilizzi le API del client `aiohttp` 3 devi anche configurare la `ClientSession` con un'istanza della configurazione di tracciamento fornita dall'SDK.

Example Client API **`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 si effettua una chiamata a un'API Web downstream, l'SDK X-Ray per Python registra un sottosegmento che contiene informazioni sulla richiesta e sulla risposta HTTP. X-Ray utilizza il sottosegmento per generare un segmento dedotto per l'API remota.

Example Sottosegmento per una chiamata HTTP a valle

```
{
  "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 dedotto per una chiamata HTTP a valle

```
{
  "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
}
```

Generazione di sottosegmenti personalizzati con X-Ray SDK per Python

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

I sottosegmenti estendono il segmento di una traccia con dettagli sul lavoro svolto per soddisfare una richiesta. Ogni volta che si effettua una chiamata con un client dotato di strumentazione, l'X-Ray SDK

registra le informazioni generate in un sottosegmento. È possibile creare sottosegmenti aggiuntivi per raggruppare altri sottosegmenti, misurare le prestazioni di una sezione di codice o registrare annotazioni e metadati.

Per gestire i sottosegmenti, utilizza i metodi `begin_subsegment` e `end_subsegment`.

Example main.py — Sottosegmento personalizzato

```
from aws_xray_sdk.core import xray_recorder

subsegment = xray_recorder.begin_subsegment('annotations')
subsegment.put_annotation('id', 12345)
xray_recorder.end_subsegment()
```

Per creare un sottosegmento personalizzato per una funzione sincrona, utilizza il decoratore `@xray_recorder.capture`. Puoi passare un nome per il sottosegmento alla funzione di acquisizione o ometterlo per utilizzare il nome della funzione.

Example main.py — Sottosegmento della funzione

```
from aws_xray_sdk.core import xray_recorder

@xray_recorder.capture('## create_user')
def create_user():
...
```

Per una funzione asincrona, utilizza il decoratore `@xray_recorder.capture_async` e passa un contesto async al registratore.

Example main.py — Sottosegmento di funzione asincrona

```
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 create un sottosegmento all'interno di un segmento o di un altro sottosegmento, X-Ray SDK for Python genera un ID per esso e registra l'ora di inizio e l'ora di fine.

Example Sottosegmento con metadati

```
"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"
    }
  },
```

Aggiungi annotazioni e metadati ai segmenti con l'SDK X-Ray per Python

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile registrare informazioni aggiuntive sulle richieste, sull'ambiente o sull'applicazione con annotazioni e metadati. È possibile aggiungere annotazioni e metadati ai segmenti creati da X-Ray SDK o ai sottosegmenti personalizzati creati dall'utente.

Le annotazioni sono coppie chiave-valore con stringhe, numeri o valori booleani. Le annotazioni sono indicizzate per essere utilizzate con le espressioni di filtro. Utilizzale per registrare i dati che desideri utilizzare per raggruppare le tracce nella console oppure per chiamare l'API `GetTraceSummaries`.

I metadati sono coppie chiave-valore che possono avere valori di qualsiasi tipo, inclusi oggetti ed elenchi, ma non sono indicizzati per essere utilizzati con le espressioni di filtro. Utilizzate i metadati per registrare dati aggiuntivi che desiderate archiviare nella traccia ma che non è necessario utilizzare con la ricerca.

Oltre ad annotazioni e metadati, sui segmenti puoi anche registrare le stringhe degli ID utente. IDs Gli utenti vengono registrati in un campo separato sui segmenti e indicizzati per essere utilizzati nella ricerca.

Sections

- Registrazione di annotazioni con X-Ray SDK per Python
- Registrazione di metadati con X-Ray SDK per Python
- Utente che registra IDs con X-Ray SDK per Python

Registrazione di annotazioni con X-Ray SDK per Python

Utilizza le annotazioni per memorizzare le informazioni su segmenti o sottosegmenti che desideri siano indicizzate per la ricerca.

Requisiti per le annotazioni

- Chiavi: la chiave per un'annotazione a raggi X può contenere fino a 500 caratteri alfanumerici. Non è possibile utilizzare spazi o simboli diversi da un punto o un punto (.)
- Valori: il valore di un'annotazione X-Ray può contenere fino a 1.000 caratteri Unicode.
- Il numero di annotazioni: è possibile utilizzare fino a 50 annotazioni per traccia.

Per registrare le annotazioni

1. Ottenere un riferimento al segmento o sottosegmento corrente da `xray_recorder`.

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_segment()
   ```

   or

   ```
   from aws_xray_sdk.core import xray_recorder
   ...
   document = xray_recorder.current_subsegment()
   ```

2. Chiamare `put_annotation` con una chiave di tipo Stringa e un valore booleano, numerico o di tipo Stringa.

```
document.put_annotation("mykey", "my value");
```

L'esempio seguente mostra come effettuare una chiamata `putAnnotation` con una chiave String che include un punto e un valore booleano, numerico o stringa.

```
document.putAnnotation("testkey.test", "my value");
```

In alternativa, puoi utilizzare il metodo `put_annotation` sul `xray_recorder`. Questo metodo memorizza le annotazioni sul sottosegmento corrente oppure, se nessun sottosegmento è aperto, sul segmento.

```
xray_recorder.put_annotation("mykey", "my value");
```

L'SDK memorizza le annotazioni come coppie chiave-valore in un oggetto `annotations` all'interno del documento di segmento. Se chiami `put_annotation` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Per trovare tracciamenti con annotazioni contenenti valori specifici, utilizza la parola chiave `annotation[key]` in un'espressione filtro.

Registrazione di metadati con X-Ray SDK per Python

> **Warning**
>
> Non aggiungere oggetti con riferimenti circolari come valori di metadati in X-Ray SDK per Python. Questi oggetti non possono essere serializzati in JSON e possono creare loop infiniti nell'SDK. Inoltre, evita di aggiungere oggetti grandi e complessi come metadati per evitare problemi di prestazioni.

Utilizza i metadati per memorizzare le informazioni su segmenti o sottosegmenti che non è necessario che siano indicizzate per la ricerca. I valori dei metadati possono essere stringhe, numeri, valori booleani o qualsiasi oggetto che possa essere serializzato in un oggetto o in un vettore JSON.

Per registrare i metadati

1. Ottenere un riferimento al segmento o sottosegmento corrente da `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

or

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. Chiama `put_metadata` con una chiave di tipo Stringa, un valore booleano, numerico, di tipo Stringa o di tipo oggetto e un namespace di tipo stringa.

```
document.put_metadata("my key", "my value", "my namespace");
```

or

Invocare `put_metadata` semplicemente con una chiave e un valore.

```
document.put_metadata("my key", "my value");
```

In alternativa, puoi utilizzare il metodo `put_metadata` sul `xray_recorder`. Questo metodo memorizza i metadati sul sottosegmento corrente oppure, se nessun sottosegmento è aperto, sul segmento.

```
xray_recorder.put_metadata("my key", "my value");
```

Se non specifichi un namespace, l'SDK utilizza `default`. Se chiami `put_metadata` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Utente che registra IDs con X-Ray SDK per Python

Registra i segmenti dell'utente IDs su richiesta per identificare l'utente che ha inviato la richiesta.

Per registrare l'utente IDs

1. Ottenere un riferimento al segmento corrente da `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

2. Chiamare `setUser` con una stringa che rappresenta l'ID dell'utente che ha inviato la richiesta.

```
document.set_user("U12345");
```

Puoi chiamare `set_user` nel tuo controller per registrare l'ID utente non appena l'applicazione inizia ad elaborare una richiesta.

Per trovare tracciamenti associati ad un ID utente, utilizza la parola chiave `user` in un'espressione filtro.

Analisi di framework Web distribuiti in ambienti serverless

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L' AWS X-Ray SDK per Python supporta la strumentazione di framework web distribuiti in applicazioni serverless. Serverless è l'architettura nativa del cloud che consente di trasferire più responsabilità operative ad AWS, aumentando agilità e innovazione.

L'architettura serverless è un modello di applicazione software che consente di creare ed eseguire applicazioni e servizi senza pensare ai server. Elimina le attività di gestione delle infrastrutture come il provisioning del server o del cluster, l'applicazione di patch, la manutenzione del sistema operativo e il provisioning della capacità. Puoi creare soluzioni serverless per pressoché ogni tipo di applicazione o servizio di back-end e tutte le operazioni necessarie per l'esecuzione e la scalabilità dell'applicazione saranno gestite in automatico.

Questo tutorial mostra come utilizzare automaticamente la strumentazione AWS X-Ray su un framework web, come Flask o Django, distribuito in un ambiente serverless. La strumentazione X-Ray dell'applicazione ti consente di visualizzare tutte le chiamate downstream effettuate, a partire da Amazon API Gateway attraverso la tua AWS Lambda funzione, e le chiamate in uscita effettuate dall'applicazione.

L'X-Ray SDK per Python supporta i seguenti framework applicativi Python:

- Flask 0.8 o versioni successive
- Django 1.0 o versioni successive

Questo tutorial sviluppa un esempio di applicazione serverless che viene distribuita su Lambda e richiamata da API Gateway. Questo tutorial utilizza Zappa per distribuire automaticamente l'applicazione su Lambda e configurare l'endpoint API Gateway.

Prerequisiti

- Zappa
- Python — Versione 2.7 o 3.6.
- AWS CLI— Verifica che il tuo AWS CLI sia configurato con l'account e Regione AWS in cui distribuirai l'applicazione.
- Pip
- Virtualenv

Fase 1: creazione di un ambiente

In questa fase, crei un ambiente virtuale usando `virtualenv` per ospitare un'applicazione.

1. Utilizzando AWS CLI, create una directory per l'applicazione. Quindi, passare alla nuova directory.

   ```
   mkdir serverless_application
   cd serverless_application
   ```

2. Successivamente, creare un ambiente virtuale all'interno della nuova directory. Utilizzare il comando seguente per attivarlo.

   ```
   # Create our virtual environment
   ```

```
virtualenv serverless_env

# Activate it
source serverless_env/bin/activate
```

3. Installa X-Ray, Flask, Zappa e la libreria Requests nel tuo ambiente.

```
# Install X-Ray, Flask, Zappa, and Requests into your environment
pip install aws-xray-sdk flask zappa requests
```

4. Aggiungere il codice dell'applicazione nella directory `serverless_application`. In questo caso, si utilizza l'esempio Hello World di Flask.

   Nella directory `serverless_application`, creare un file denominato `my_app.py`. Quindi utilizzare un editor di testo per aggiungere i seguenti comandi. Questa applicazione analizza la libreria delle richieste, applica le patch al middleware dell'applicazione Flask e apre l'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
```

Fase 2: creazione e distribuzione di un ambiente Zappa

In questo passaggio utilizzerai Zappa per configurare automaticamente un endpoint API Gateway e poi distribuirlo su Lambda.

1. Inizializzare Zappa dalla directory `serverless_application`. Per questo esempio vengono utilizzate le impostazioni predefinite, ma se si desidera personalizzare le preferenze, è possibile visualizzare le istruzioni di configurazione di Zappa.

```
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. Abilita X-Ray. Aprire il file `zappa_settings.json` e verificare che sia simile all'esempio.

```
{
    "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. Aggiungere la voce `"xray_tracing": true` al file di configurazione.

```
{
    "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. Distribuire l'applicazione. Questo configura automaticamente l'endpoint API Gateway e carica il codice su Lambda.

```
zappa deploy
```

```
...
Deploying API Gateway..
Deployment complete!: https://**********.execute-api.us-west-2.amazonaws.com/dev
```

Fase 3: Abilitare il tracciamento a raggi X per API Gateway

In questo passaggio interagirai con la console API Gateway per abilitare il tracciamento X-Ray.

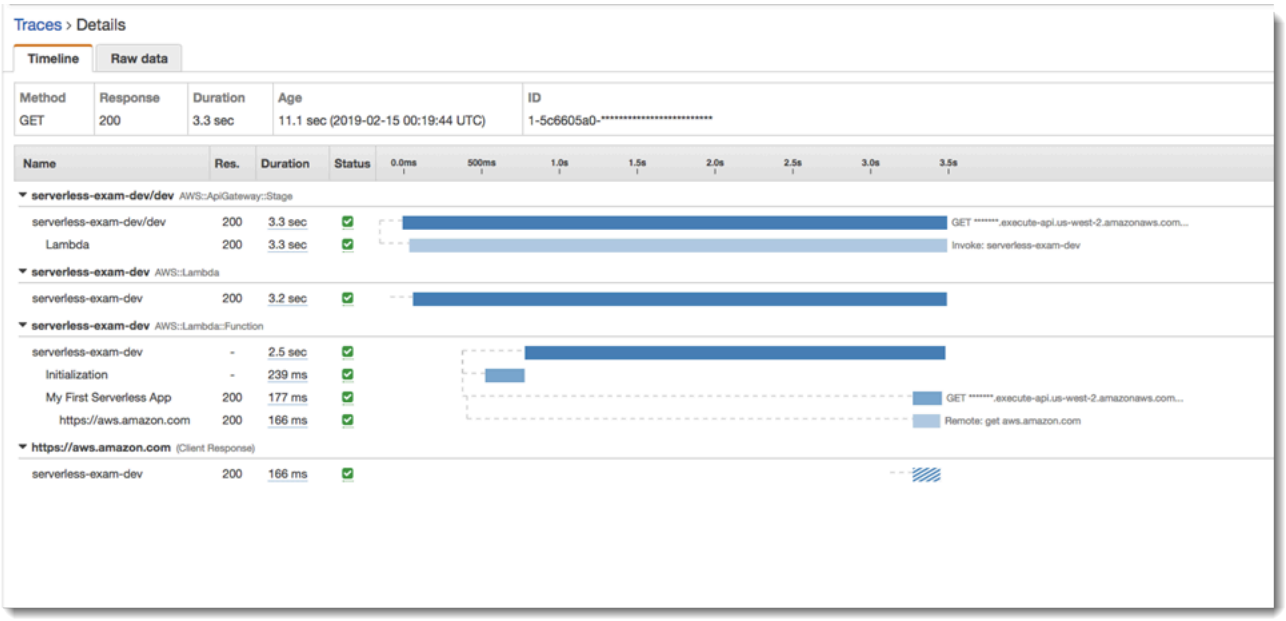
1. Accedi Console di gestione AWS e apri la console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway/.
2. Trovare l'API generata di recente. Deve avere un formato analogo a `serverless-exam-dev`.
3. Scegliere Stages (Fasi).
4. Scegliere il nome della fase di distribuzione. Il valore predefinito è `dev`.
5. Nella scheda Log/Tracciamento, selezionare la casella Enable X-Ray Tracing (Abilita tracciamento X-Ray).
6. Seleziona Salva modifiche.
7. Accedere all'endpoint nel browser. Se è stata utilizzata l'applicazione di esempio `Hello World`, viene visualizzato quanto segue.

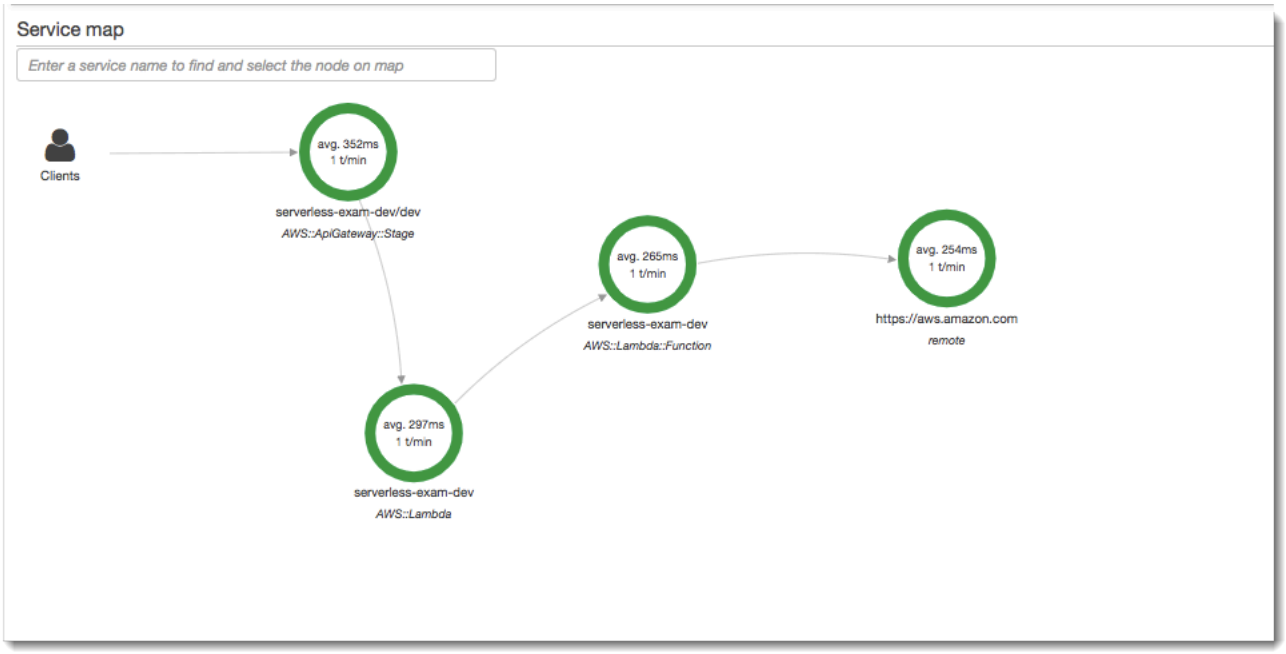
```
"Hello, World: https://aws.amazon.com/"
```

Fase 4: visualizzazione della traccia creata

In questo passaggio interagirai con la console X-Ray per visualizzare la traccia creata dall'applicazione di esempio. Per una spiegazione passo per passo sull'analisi di traccia, consulta Visualizzazione della mappa del servizio.

1. Accedi a Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Visualizza i segmenti generati da API Gateway, dalla funzione Lambda e dal contenitore Lambda.
3. Nel segmento della funzione Lambda, visualizza un sottosegmento denominato `My First Serverless App` È seguito da un secondo sottosegmento denominato `https://aws.amazon.com`.
4. Durante l'inizializzazione, Lambda potrebbe anche generare un terzo sottosegmento denominato. `initialization`





Fase 5: rimozione

Termina sempre le risorse che non usi più per evitare l'accumulo di costi imprevisti. Come dimostra questo tutorial, gli strumenti come Zappa semplificano la ridistribuzione serverless.

Per rimuovere l'applicazione da Lambda, API Gateway e Amazon S3, esegui il comando seguente nella directory del progetto utilizzando il. AWS CLI

```
zappa undeploy dev
```

Fasi successive

Aggiungi altre funzionalità alla tua applicazione aggiungendo AWS client e strumentandoli con X-Ray. Scopri di più sulle opzioni di elaborazione serverless su Serverless on. AWS

Utilizzo di .NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Esistono due modi per strumentare un'applicazione.NET per inviare tracce a X-Ray:

- AWS Distro for OpenTelemetry .NET: una AWS distribuzione che fornisce un set di librerie open source per l'invio di metriche e tracce correlate a più soluzioni di AWS monitoraggio tra cui Amazon AWS X-Ray e Amazon OpenSearch Service CloudWatch, tramite AWS Distro for Collector. OpenTelemetry
- AWS X-Ray SDK for .NET: un insieme di librerie per la generazione e l'invio di tracce a X-Ray tramite il demone X-Ray.

Per ulteriori informazioni, consulta Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs.

AWS Distro per.NET OpenTelemetry

Con AWS Distro for OpenTelemetry .NET, puoi strumentare le tue applicazioni una sola volta e inviare metriche e tracce correlate a più soluzioni di AWS monitoraggio tra cui Amazon CloudWatch AWS X-Ray e Amazon Service. OpenSearch L'utilizzo di X-Ray con AWS Distro for OpenTelemetry richiede due componenti: un OpenTelemetry SDK abilitato per l'uso con X-Ray e AWS Distro for OpenTelemetry Collector abilitato per l'uso con X-Ray.

Per iniziare, consulta la documentazione di Distro for .NET.AWS OpenTelemetry

Per ulteriori informazioni sull'utilizzo di AWS Distro for OpenTelemetry AWS X-Ray e altro Servizi AWS, consulta AWS Distro for OpenTelemetry o Distro for DocumentationAWS . OpenTelemetry

Per ulteriori informazioni sul supporto e l'utilizzo della lingua, vedere AWS Observability on. GitHub

AWS X-Ray SDK per.NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

X-Ray SDK for .NET è una libreria per la strumentazione delle applicazioni Web C# .NET, delle applicazioni Web.NET Core e delle funzioni.NET Core. AWS Lambda Fornisce classi e metodi per generare e inviare dati di traccia al demone X-Ray. Ciò include informazioni sulle richieste in entrata fornite dall'applicazione e sulle chiamate effettuate dall'applicazione ai database downstream Servizi AWS, HTTP Web APIs e SQL.

> **Note**
>
> X-Ray SDK for .NET è un progetto open source. Puoi seguire il progetto e inviare problemi e richieste pull su GitHub: github. com/aws/aws- xray-sdk-dotnet

In caso di applicazioni web, per tracciare le richieste in entrata inizia aggiungendo un gestore di messaggi nella configurazione web. Il gestore dei messaggi crea un segmento per ogni richiesta tracciata e completa il segmento quando viene inviata la risposta. Fino a che il segmento è aperto puoi usare i metodi del client dell'SDK per aggiungere informazioni al segmento e creare sottosegmenti per tracciare le chiamate a valle. L'SDK, inoltre, registra automaticamente le eccezioni sollevate dall'applicazione per il tempo durante il quale il segmento è aperto.

Per le funzioni Lambda richiamate da un'applicazione o un servizio strumentato, Lambda legge l'intestazione di tracciamento e traccia automaticamente le richieste campionate. Per altre funzioni, puoi configurare Lambda per campionare e tracciare le richieste in arrivo. In entrambi i casi, Lambda crea il segmento e lo fornisce all'X-Ray SDK.

> **Note**
>
> Su Lambda, l'SDK X-Ray è opzionale. Se non lo usi nella tua funzione, la mappa dei servizi includerà comunque un nodo per il servizio Lambda e uno per ogni funzione Lambda. Aggiungendo l'SDK, puoi utilizzare il codice della funzione per aggiungere sottosegmenti al segmento di funzione registrato da Lambda. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Successivamente, usa l'X-Ray SDK for .NET per strumentare i tuoi clienti. AWS SDK per .NET Ogni volta che effettui una chiamata a un downstream Servizio AWS o a una risorsa con un client dotato di strumentazione, l'SDK registra le informazioni sulla chiamata in un sottosegmento. AWS i servizi e le risorse a cui accedi all'interno dei servizi vengono visualizzati come nodi a valle sulla mappa di traccia per aiutarti a identificare errori e problemi di limitazione sulle singole connessioni.

L'X-Ray SDK for .NET fornisce anche la strumentazione per le chiamate downstream ai database HTTP Web e SQL. APIs Il metodo di estensione `GetResponseTraced` di `System.Net.HttpWebRequest` traccia le chiamate HTTP in uscita. È possibile utilizzare la versione X-Ray SDK for .NET di per `SqlCommand` strumentare le query SQL.

Dopo aver iniziato a utilizzare l'SDK, personalizzane il comportamento configurando il registratore e il gestore dei messaggi. Puoi aggiungere dei plugin per memorizzare i dati sulle risorse di elaborazione sulle quali è eseguita la tua applicazione, personalizzare il comportamento di campionamento definendo regole di campionatura e impostare il livello di log per visualizzare più o meno informazioni generate dall'SDK nei log dell'applicazione.

Registra ulteriori informazioni sulle richieste e sull'attività svolta dalla tua applicazione in annotazioni e metadati. Le annotazioni sono semplici coppie chiave-valore indicizzati per l'uso con espressioni filtro, in modo da poter cercare tracce che contengono dati specifici. Le immissioni di metadati sono meno restrittive e possono registrare interi oggetti e matrici, ovvero tutto ciò che può essere serializzato in JSON.

> **Annotazioni e metadata**
>
> Le annotazioni e i metadati sono testo arbitrario che aggiungi ai segmenti con X-Ray SDK. Le annotazioni vengono indicizzate per essere utilizzate con le espressioni di filtro. I metadati non sono indicizzati, ma possono essere visualizzati nel segmento non elaborato con la

console X-Ray o l'API. Chiunque conceda l'accesso in lettura a X-Ray può visualizzare questi dati.

Quando disponi di una notevole quantità di client analizzati nel tuo codice, un singolo segmento per la richiesta può contenere un numero elevato di sottosegmenti, uno per ciascuna delle chiamate effettuate con un client analizzato. Puoi organizzare e raggruppare i sottosegmenti inglobando le chiamate del client sottosegmenti personalizzati. Puoi creare un sottosegmento personalizzato per un'intera funzione o qualsiasi porzione di codice, e memorizzare metadati e annotazioni sul sottosegmento invece di scrivere tutto all'interno del segmento padre.

Per la documentazione di riferimento sulle classi e sui metodi dell'SDK, consulta quanto indicato qui di seguito:

- AWS X-Ray Riferimento all'API SDK for .NET
- AWS X-Ray Guida di riferimento all'API SDK for .NET Core

Lo stesso pacchetto supporta sia .NET che .NET Core, ma le classi utilizzate sono diverse. Gli esempi in questo capitolo sono collegati alla guida di riferimento per le API .NET a meno che la classe non sia specifica per .NET Core.

Requisiti

X-Ray SDK for .NET richiede .NET Framework 4.5 o versione successiva e. AWS SDK per .NET

In caso di applicazioni e funzioni .NET Core, l'SDK richiede .NET Core 2.0 o versioni successive.

Aggiungere l'X-Ray SDK for .NET all'applicazione

NuGet Usalo per aggiungere l'X-Ray SDK for .NET alla tua applicazione.

Per installare X-Ray SDK per .NET con il gestore di NuGet pacchetti in Visual Studio

1. Scegli Tools, NuGet Package Manager, Manage NuGet Packages for Solution.
2. Cerca AWSXRayRecorder.
3. Scegliere il pacchetto e quindi scegliere Install (Installa).

Gestione delle dipendenze

L'X-Ray SDK for .NET è disponibile presso Nuget. Installa l'SDK utilizzando il gestore di pacchetti:

```
Install-Package AWSXRayRecorder -Version 2.10.1
```

Il pacchetto `AWSXRayRecorder v2.10.1` nuget ha le seguenti dipendenze:

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)
```

Per ulteriori dettagli sulla gestione delle dipendenze, consulta la documentazione di Microsoft sulla dipendenza da Nuget e sulla risoluzione delle dipendenze da Nuget.

Configurazione dell'X-Ray SDK per.NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile configurare X-Ray SDK for .NET con plug-in per includere informazioni sul servizio su cui viene eseguita l'applicazione, modificare il comportamento di campionamento predefinito o aggiungere regole di campionamento che si applicano alle richieste a percorsi specifici.

Per le applicazioni web .NET, aggiungi le chiavi alla sezione `appSettings` del file `Web.config`.

Example Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Per .NET Core, crea un file denominato `appsettings.json` con una chiave di primo livello denominata `XRay`.

Example.NET appsettings.json

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin",
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

Quindi, nel codice dell'applicazione, create un oggetto di configurazione e utilizzatelo per inizializzare il registratore X-Ray. Esegui questa operazione prima di inizializzare il registratore.

Example.NET Core Program.cs — Configurazione del registratore

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Se stai analizzando un'applicazione web .NET Core, puoi anche inoltrare l'oggetto di configurazione al metodo `UseXRay` quando configuri il gestore dei messaggi. Per le funzioni Lambda, usa il `InitializeInstance` metodo illustrato sopra.

Per ulteriori informazioni sull'API di configurazione.NET Core, consulta Configurare un'app ASP.NET Core su docs.microsoft.com.

Sections

- Plugin
- Regole di campionamento
- Logging (.NET)
- Logging (.NET Core)
- Variabili di ambiente

Plugin

Utilizza i plug-in per aggiungere i dati sul servizio che ospita l'applicazione.

Plugin

- Amazon EC2 : `EC2Plugin` aggiunge l'ID dell'istanza, la zona di disponibilità e il gruppo di CloudWatch log.
- Elastic `ElasticBeanstalkPlugin` Beanstalk: aggiunge il nome dell'ambiente, l'etichetta della versione e l'ID di distribuzione.
- Amazon ECS: `ECSPlugin` aggiunge l'ID del contenitore.

Per utilizzare un plug-in, configura il client X-Ray SDK for .NET aggiungendo l'impostazione. `AWSXRayPlugins` Se si applicano più plugin all'applicazione, specificarli tutti nella stessa impostazione, separati da virgole.

Example Web.config - Plugin

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin,ElasticBeanstalkPlugin"/>
  </appSettings>
</configuration>
```

Example.NET Core appsettings.json — Plugin

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin,ElasticBeanstalkPlugin"
  }
}
```

Regole di campionamento

L'SDK utilizza le regole di campionamento definite nella console X-Ray per determinare quali richieste registrare. La regola predefinita tiene traccia della prima richiesta ogni secondo e del cinque percento di eventuali richieste aggiuntive su tutti i servizi che inviano tracce a X-Ray. Crea regole aggiuntive nella console X-Ray per personalizzare la quantità di dati registrati per ciascuna delle tue applicazioni.

L'SDK applica le regole personalizzate nell'ordine in cui sono definite. Se una richiesta corrisponde a più regole personalizzate, l'SDK applica solo la prima regola.

> **Note**
>
> Se l'SDK non riesce a contattare X-Ray per ottenere le regole di campionamento, torna a una regola locale predefinita della prima richiesta ogni secondo e del cinque percento di eventuali richieste aggiuntive per host. Ciò può verificarsi se l'host non dispone dell'autorizzazione per chiamare sampling APIs o non riesce a connettersi al demone X-Ray, che funge da proxy TCP per le chiamate API effettuate dall'SDK.

Puoi anche configurare l'SDK per caricare le regole di campionamento da un documento JSON. L'SDK può utilizzare le regole locali come backup per i casi in cui il campionamento a raggi X non è disponibile o utilizzare esclusivamente regole locali.

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
  }
}
```

Questo esempio definisce una regola personalizzata e una regola predefinita. La regola personalizzata applica una frequenza di campionamento del cinque percento senza alcun numero minimo di richieste da tracciare per i percorsi. `/api/move/` La regola predefinita tiene traccia della prima richiesta ogni secondo e del 10% delle richieste aggiuntive.

Lo svantaggio della definizione locale delle regole è che l'obiettivo fisso viene applicato da ciascuna istanza del registratore in modo indipendente, anziché essere gestito dal servizio X-Ray. Man mano che si installano più host, la tariffa fissa si moltiplica, rendendo più difficile il controllo della quantità di dati registrati.

Sì AWS Lambda, non è possibile modificare la frequenza di campionamento. Se la funzione viene chiamata da un servizio strumentato, le chiamate che hanno generato richieste campionate da quel servizio verranno registrate da Lambda. Se il tracciamento attivo è abilitato e non è presente alcuna intestazione di tracciamento, Lambda prende la decisione di campionamento.

Per configurare le regole di backup, chiedi a X-Ray SDK for .NET di caricare le regole di campionamento da un file con l'impostazione. `SamplingRuleManifest`

Example.NET Web.config - regole di campionamento

```
<configuration>
  <appSettings>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Example.NET Core appsettings.json — Regole di campionamento

```
{
  "XRay": {
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

Per utilizzare solo regole locali, crea un registratore con una `LocalizedSamplingStrategy`. Se si dispone di regole di backup configurate, rimuovere tale configurazione.

Example.NET global.asax: regole di campionamento locali

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("samplingrules.json")).Build();
AWSXRayRecorder.InitializeInstance(recorder: recorder);
```

Example.NET Core Program.cs: regole di campionamento locali

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("sampling-rules.json")).Build();
AWSXRayRecorder.InitializeInstance(configuration,recorder);
```

Logging (.NET)

L'X-Ray SDK for .NET utilizza lo stesso meccanismo di registrazione di. AWS SDK per .NET Se hai già configurato l'applicazione per registrare l' AWS SDK per .NET output, la stessa configurazione si applica all'output di X-Ray SDK for .NET.

Per configurare il log, aggiungi una sezione di configurazione denominata `aws` al tuo file `App.config` o al tuo file `Web.config`.

Example Web.config - Generazione dei log

```
...
<configuration>
  <configSections>
    <section name="aws" type="Amazon.AWSSection, AWSSDK.Core"/>
  </configSections>
  <aws>
    <logging logTo="Log4Net"/>
  </aws>
</configuration>
```

Per ulteriori informazioni, consulta Configurazione dell'applicazione AWS SDK per .NET nella Guida per gli sviluppatori di AWS SDK per .NET .

Logging (.NET Core)

X-Ray SDK for .NET utilizza le stesse opzioni di registrazione di. AWS SDK per .NET Per configurare la registrazione per le applicazioni.NET Core, passa l'opzione di registrazione al metodo. `AWSXRayRecorder.RegisterLogger`

Ad esempio, per utilizzare log4net, è necessario creare un file di configurazione che definisca il logger, formato di output e la posizione del file.

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>
```

Quindi creare il logger e applicare la configurazione al codice del programma.

Example.NET Core Program.cs — Registrazione

```
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)
  {
  ...
  }
}
```

Per ulteriori informazioni sulla configurazione di log4net, vedere Configurazione su logging.apache.org.

Variabili di ambiente

È possibile utilizzare le variabili di ambiente per configurare l'X-Ray SDK for .NET. L'SDK supporta le seguenti variabili.

- `AWS_XRAY_TRACING_NAME`— Imposta un nome di servizio che l'SDK utilizza per i segmenti. Sostituisce il nome del servizio impostato sulla strategia di denominazione dei segmenti del filtro servlet.
- `AWS_XRAY_DAEMON_ADDRESS`— Imposta l'host e la porta del demone X-Ray. Per impostazione predefinita, l'SDK utilizza `127.0.0.1:2000` sia i dati di traccia (UDP) che il campionamento (TCP). Utilizzate questa variabile se avete configurato il demone per l'ascolto su una porta diversa o se è in esecuzione su un host diverso.

  Formato

  - Stessa porta — *`address`*`:`*`port`*
  - Porte diverse: `tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_CONTEXT_MISSING`— Imposta `RUNTIME_ERROR` per generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.

  Valori validi

  - `RUNTIME_ERROR`— Genera un'eccezione di runtime.
  - `LOG_ERROR`— Registra un errore e continua (impostazione predefinita).
  - `IGNORE_ERROR`— Ignora l'errore e continua.

  Gli errori relativi a segmenti o sottosegmenti mancanti possono verificarsi quando si tenta di utilizzare un client con strumenti nel codice di avvio che viene eseguito quando non è aperta alcuna richiesta o nel codice che genera un nuovo thread.

Strumentazione delle richieste HTTP in entrata con l'X-Ray SDK for .NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e

Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi utilizzare l'SDK X-Ray per tracciare le richieste HTTP in entrata che la tua applicazione serve su un'istanza EC2 in Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Utilizza un gestore di messaggi per analizzare le richieste HTTP in ingresso. Quando aggiungete il gestore di messaggi X-Ray all'applicazione, l'X-Ray SDK for .NET crea un segmento per ogni richiesta campionata. Questo segmento include durata, metodo e conclusione della richiesta HTTP. Analisi ulteriori creano sottosegmenti associati a questo segmento.

> **Note**
>
> Per quanto riguarda AWS Lambda le funzioni, Lambda crea un segmento per ogni richiesta campionata. Per ulteriori informazioni, consulta AWS Lambda e AWS X-Ray.

Ogni segmento ha un nome che identifica l'applicazione nella mappa dei servizi. Il segmento può essere denominato staticamente oppure è possibile configurare l'SDK per denominarlo dinamicamente in base all'intestazione dell'host nella richiesta in entrata. La denominazione dinamica consente di raggruppare le tracce in base al nome di dominio nella richiesta e di applicare un nome predefinito se il nome non corrisponde a uno schema previsto (ad esempio, se l'intestazione dell'host è falsificata).

> **Richieste inoltrate**
>
> Se un sistema di bilanciamento del carico o un altro intermediario inoltra una richiesta all'applicazione, X-Ray prende l'IP del client dall'`X-Forwarded-For`intestazione della richiesta anziché dall'IP di origine nel pacchetto IP. L'IP del client registrato per una richiesta inoltrata può essere falsificato, quindi non dovrebbe essere considerato attendibile.

Il gestore dei messaggi crea un segmento per ogni richiesta in entrata con un blocco `http` che contiene le informazioni riportate qui di seguito:

- Metodo HTTP: GET, POST, PUT, DELETE, ecc.

- Indirizzo client: l'indirizzo IP del client che ha inviato la richiesta.
- Codice di risposta: il codice di risposta HTTP per la richiesta completata.
- Tempistica: l'ora di inizio (quando è stata ricevuta la richiesta) e l'ora di fine (quando è stata inviata la risposta).
- Agente utente: il `user-agent` codice della richiesta.
- Lunghezza del contenuto: il `content-length` risultato della risposta.

Sections

- Analisi delle richieste in entrata (.NET)
- Analisi delle richieste in entrata (.NET Core)
- Configurazione di una strategia di denominazione dei segmenti

Analisi delle richieste in entrata (.NET)

Per analizzare le richieste elaborate dalla tua applicazione, chiama `RegisterXRay` nel metodo `Init` del tuo file `global.asax`.

Example global.asax - Gestore messaggi

```
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");
    }
  }
}
```

Analisi delle richieste in entrata (.NET Core)

Per le richieste di strumenti servite dalla tua applicazione, chiama `UseXRay` method prima di qualsiasi altro middleware nel `Configure` metodo della tua classe Startup, poiché idealmente il

middleware X-Ray dovrebbe essere il primo middleware a elaborare la richiesta e l'ultimo middleware a elaborare la risposta nella pipeline.

> **Note**
>
> Per .NET Core 2.0, se hai un `UseExceptionHandler` metodo nell'applicazione, assicurati di chiamare dopo metodo per assicurarti che le eccezioni vengano registrate. `UseXRay` `UseExceptionHandler`

Example Startup.cs

<caption>.NET Core 2.1 and above</caption>

```
using Microsoft.AspNetCore.Builder;

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {
    app.UseXRay("MyApp");
    // additional middleware
    ...
  }
```

<caption>.NET Core 2.0</caption>

```
using Microsoft.AspNetCore.Builder;

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {
    app.UseExceptionHandler("/Error");
    app.UseXRay("MyApp");
    // additional middleware
    ...
  }
```

Il metodo `UseXRay` può anche ricevere un oggetto di configurazione come secondo argomento.

```
app.UseXRay("MyApp", configuration);
```

Configurazione di una strategia di denominazione dei segmenti

AWS X-Ray utilizza un nome di servizio per identificare l'applicazione e distinguerla dalle altre applicazioni, database, risorse esterne APIs e AWS risorse utilizzate dall'applicazione. Quando X-Ray SDK genera segmenti per le richieste in entrata, registra il nome del servizio dell'applicazione nel campo del nome del segmento.

L'X-Ray SDK può denominare i segmenti dopo il nome host nell'intestazione della richiesta HTTP. Tuttavia, questa intestazione può essere falsificata, il che potrebbe causare nodi imprevisti nella mappa dei servizi. Per evitare che l'SDK nomini i segmenti in modo errato a causa di richieste con intestazioni host contraffatte, è necessario specificare un nome predefinito per le richieste in entrata.

Se la tua applicazione soddisfa le richieste per più domini, puoi configurare l'SDK in modo che utilizzi una strategia di denominazione dinamica che rifletta questo aspetto nei nomi dei segmenti. Una strategia di denominazione dinamica consente all'SDK di utilizzare il nome host per le richieste che corrispondono a uno schema previsto e di applicare il nome predefinito alle richieste che non lo fanno.

Ad esempio, potresti avere una singola applicazione che serve le richieste a tre sottodomini:, e. `www.example.com` `api.example.com` `static.example.com` È possibile utilizzare una strategia di denominazione dinamica con lo schema `*.example.com` per identificare i segmenti per ogni sottodominio con un nome diverso, ottenendo tre nodi di servizio sulla mappa dei servizi. Se l'applicazione riceve richieste con un nome host che non corrisponde allo schema, sulla mappa dei servizi verrà visualizzato un quarto nodo con un nome di fallback specificato dall'utente.

Per utilizzare lo stesso nome per tutti i segmenti della richiesta, specifica il nome della tua applicazione quando inizializzi il gestore dei messaggi, come illustrato nella sezione precedente. Ciò ha lo stesso effetto di creare una `FixedSegmentNamingStrategy` e passarla al metodo `RegisterXRay`.

```
AWSXRayASPNET.RegisterXRay(this, new FixedSegmentNamingStrategy("MyApp"));
```

> **Note**
>
> È possibile sovrascrivere il nome di servizio predefinito definito nel codice con la `AWS_XRAY_TRACING_NAME` variabile di ambiente.

Una strategia di denominazione dinamica definisce un modello al quale devono corrispondere i nomi degli host e un nome di default per l'utilizzo qualora il nome dell'host nella richiesta HTTP non corrisponda al modello. Per denominare i segmenti in modo dinamico, è necessario creare una `DynamicSegmentNamingStrategy` e passarla al metodo `RegisterXRay`.

```
AWSXRayASPNET.RegisterXRay(this, new DynamicSegmentNamingStrategy("MyApp",
 "*.example.com"));
```

Tracciamento delle chiamate AWS SDK con X-Ray SDK for .NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate per Servizi AWS archiviare dati, scrivere in una coda o inviare notifiche, X-Ray SDK for .NET tiene traccia delle chiamate downstream in sottosegmenti. Le risorse tracciate Servizi AWS e a cui accedi all'interno di tali servizi (ad esempio, un bucket Amazon S3 o una coda Amazon SQS) vengono visualizzate come nodi downstream sulla mappa di traccia nella console X-Ray.

Puoi strumentare tutti i tuoi AWS SDK per .NET clienti chiamando prima di crearli. `RegisterXRayForAllServices`

Example SampleController.cs - Strumentazione client DynamoDB

```
using Amazon;
using Amazon.Util;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.AwsSdk;

namespace SampleEBWebApplication.Controllers
{
```

```
  public class SampleController : ApiController
  {
    AWSSDKHandler.RegisterXRayForAllServices();
    private static readonly Lazy<AmazonDynamoDBClient> LazyDdbClient = new
Lazy<AmazonDynamoDBClient>(() =>
    {
      var client = new AmazonDynamoDBClient(EC2InstanceMetadata.Region ??
RegionEndpoint.USEast1);
      return client;
    });
```

Per analizzare il client di alcuni servizi e non di altri, chiama `RegisterXRay` invece di `RegisterXRayForAllServices`. Sostituisci il testo evidenziato con il nome dell'interfaccia client del servizio.

```
AWSSDKHandler.RegisterXRay<IAmazonDynamoDB>()
```

Per tutti i servizi, puoi vedere il nome dell'API richiamata nella console X-Ray. Per un sottoinsieme di servizi, l'SDK X-Ray aggiunge informazioni al segmento per fornire una maggiore granularità nella mappa dei servizi.

Ad esempio, quando si effettua una chiamata con un client DynamoDB con strumentazione, l'SDK aggiunge il nome della tabella al segmento per le chiamate destinate a una tabella. Nella console, ogni tabella appare come un nodo separato nella mappa dei servizi, con un nodo DynamoDB generico per le chiamate che non hanno come destinazione una tabella.

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento

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

```
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Quando si accede alle risorse con nome, le chiamate ai seguenti servizi creano ulteriori nodi della mappa del servizio. Le chiamate che non sono hanno come obiettivo risorse specifiche creano un nodo generico per il servizio.

- Amazon DynamoDB: nome della tabella
- Amazon Simple Storage Service: nome del bucket e della chiave
- Amazon Simple Queue Service: nome della coda

Tracciamento delle chiamate ai servizi Web HTTP downstream con X-Ray SDK for .NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate a microservizi o a HTTP pubblico APIs, è possibile utilizzare il metodo di estensione di X-Ray SDK for .NET `GetResponseTraced` `System.Net.HttpWebRequest` per strumentare tali chiamate e aggiungere l'API al grafico del servizio come servizio 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();
}
```

Per le chiamate asincrone, utilizza `GetAsyncResponseTraced`.

```
request.GetAsyncResponseTraced();
```

Se utilizzi `system.net.http.httpclient`, usa il gestore della delegazione `HttpClientXRayTracingHandler` per memorizzare le chiamate.

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 si effettua una chiamata a un'API Web downstream, l'X-Ray SDK for .NET registra un sottosegmento con informazioni sulla richiesta e sulla risposta HTTP. X-Ray utilizza il sottosegmento per generare un segmento dedotto per l'API.

Example Sottosegmento per una chiamata HTTP a valle

```
{
  "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 dedotto per una chiamata HTTP a valle

```
{
  "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
}
```

Tracciamento delle query SQL con X-Ray SDK for .NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

L'X-Ray SDK for .NET fornisce una classe wrapper`System.Data.SqlClient.SqlCommand`, denominata`TraceableSqlCommand`, che è possibile utilizzare al posto di. `SqlCommand` Puoi inizializzare un comando SQL con la classe `TraceableSqlCommand`.

Tracciamento di query SQL con metodi sincroni e asincroni

I seguenti esempi mostrano come utilizzare `TraceableSqlCommand` per tracciare automaticamente query SQL Server in modo sincrono e asincrono.

Example **`Controller.cs`** - Analisi client SQL (sincrono)

```
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();
  }
}
```

Puoi eseguire la query in modo asincrono utilizzando il metodo `ExecuteReaderAsync`.

Example **`Controller.cs`** - Analisi client SQL (Asincrono)

```
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();
```

```
  }
}
```

Raccolta di query SQL effettuate su SQL Server

Puoi abilitare l'acquisizione di `SqlCommand.CommandText` come parte del sottosegmento creato dalla query SQL. `SqlCommand.CommandText` viene visualizzato come il campo `sanitized_query` nel sottosegmento JSON. Per impostazione predefinita, questa caratteristica è disabilitata per motivi di sicurezza.

> **Note**
>
> Non abilitare la caratteristica di raccolta se si includono informazioni sensibili come testo in chiaro nelle query SQL.

Puoi abilitare la raccolta di query SQL in due modi:

- Imposta la proprietà `CollectSqlQueries` su `true` nella configurazione globale dell'applicazione.
- Imposta il parametro `collectSqlQueries` nell'istanza `TraceableSqlCommand` su `true` per raccogliere le chiamate all'interno dell'istanza.

Abilita la proprietà globale CollectSqlQueries

I seguenti esempi mostrano come abilitare la proprietà `CollectSqlQueries` per .NET e .NET Core.

.NET

Per impostare la proprietà `CollectSqlQueries` su `true` nella configurazione globale dell'applicazione in .NET, modifica la `appsettings` del file `Web.config` o `App.config`, come mostrato.

Example **`App.config`**Oppure**`Web.config`**: abilita la raccolta di query SQL a livello globale

```
<configuration>
<appSettings>
    <add key="CollectSqlQueries" value="true">
</appSettings>
</configuration>
```

.NET Core

Per impostare la `CollectSqlQueries` proprietà su `true` nella configurazione globale dell'applicazione in.NET Core, modifica il `appsettings.json` file con la chiave X-Ray, come illustrato.

Example **`appsettings.json`**— Abilita la raccolta di query SQL a livello globale

```
{
  "XRay": {
    "CollectSqlQueries":"true"
  }
}
```

Abilita il collectSqlQueries parametro

Puoi impostare il parametro `collectSqlQueries` nell'istanza `TraceableSqlCommand` su per `true` per raccogliere il testo della query SQL per le query SQL Server eseguite utilizzando tale istanza. L'impostazione del parametro su `false` disabilita la caratteristica `CollectSqlQuery` per l'istanza `TraceableSqlCommand`.

> **Note**
>
> Il valore di `collectSqlQueries` nell'istanza `TraceableSqlCommand` sostituisce il valore impostato nella configurazione globale della proprietà `CollectSqlQueries`.

Example Esempio**`Controller.cs`**: abilita la raccolta di query SQL per l'istanza

```
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();
  }
}
```

Creazione di sottosegmenti aggiuntivi

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

I sottosegmenti estendono il segmento di una traccia con dettagli sul lavoro svolto per soddisfare una richiesta. Ogni volta che si effettua una chiamata con un client dotato di strumentazione, l'X-Ray SDK registra le informazioni generate in un sottosegmento. È possibile creare sottosegmenti aggiuntivi per raggruppare altri sottosegmenti, misurare le prestazioni di una sezione di codice o registrare annotazioni e metadati.

Per gestire i sottosegmenti, utilizza i metodi `BeginSubsegment` e `EndSubsegment`. Esegui qualsiasi attività nel sottosegmento in un blocco `try` e utilizza `AddException` per tracciare le eccezioni. Chiama `EndSubsegment` in un blocco `finally` per assicurarti che il sottosegmento venga chiuso.

Example Controller.cs — Sottosegmento personalizzato

```
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
try
{
  DoWork();
}
catch (Exception e)
{
  AWSXRayRecorder.Instance.AddException(e);
}
finally
```

```
{
  AWSXRayRecorder.Instance.EndSubsegment();
}
```

Quando si crea un sottosegmento all'interno di un segmento o di un altro sottosegmento, X-Ray SDK for .NET genera un ID per tale sottosegmento e registra l'ora di inizio e l'ora di fine.

Example Sottosegmento con metadati

```
"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"
    }
  },
```

Aggiungi annotazioni e metadati ai segmenti con X-Ray SDK for .NET

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile registrare informazioni aggiuntive sulle richieste, sull'ambiente o sull'applicazione con annotazioni e metadati. È possibile aggiungere annotazioni e metadati ai segmenti creati da X-Ray SDK o ai sottosegmenti personalizzati creati dall'utente.

Le annotazioni sono coppie chiave-valore con stringhe, numeri o valori booleani. Le annotazioni sono indicizzate per essere utilizzate con le espressioni di filtro. Utilizzale per registrare i dati che desideri utilizzare per raggruppare le tracce nella console oppure per chiamare l'API `GetTraceSummaries`.

I metadati sono coppie chiave-valore che possono avere valori di qualsiasi tipo, inclusi oggetti ed elenchi, ma non sono indicizzati per essere utilizzati con le espressioni di filtro. Utilizzate i metadati per registrare dati aggiuntivi che desiderate archiviare nella traccia ma che non è necessario utilizzare con la ricerca.

Sections

- Registrazione delle annotazioni con X-Ray SDK for .NET
- Registrazione di metadati con X-Ray SDK for .NET

Registrazione delle annotazioni con X-Ray SDK for .NET

Utilizza le annotazioni per memorizzare le informazioni su segmenti o sottosegmenti che desideri siano indicizzate per la ricerca.

Quanto segue è necessario per tutte le annotazioni in X-Ray:

Requisiti per le annotazioni

- Chiavi: la chiave per un'annotazione a raggi X può contenere fino a 500 caratteri alfanumerici. Non è possibile utilizzare spazi o simboli diversi da un punto o un punto (.)
- Valori: il valore di un'annotazione X-Ray può contenere fino a 1.000 caratteri Unicode.
- Il numero di annotazioni: è possibile utilizzare fino a 50 annotazioni per traccia.

Per registrare annotazioni al di fuori di una funzione AWS Lambda

1. Procurarsi un'istanza di `AWSXRayRecorder`.

   ```
   using Amazon.XRay.Recorder.Core;
   ...
   AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
   ```

2. Chiamare `addAnnotation` con una chiave di tipo Stringa e un valore booleano, Int32, Int64, Double o Stringa.

   ```
   recorder.AddAnnotation("mykey", "my value");
   ```

   L'esempio seguente mostra come effettuare una chiamata `putAnnotation` con una chiave String che include un punto e un valore booleano, numerico o stringa.

```
document.putAnnotation("testkey.test", "my value");
```

Per registrare annotazioni all'interno di una funzione AWS Lambda

Sia i segmenti che i sottosegmenti all'interno di una funzione Lambda sono gestiti dall'ambiente di runtime Lambda. Se desideri aggiungere un'annotazione a un segmento o sottosegmento all'interno di una funzione Lambda, devi fare quanto segue:

1. Crea il segmento o il sottosegmento all'interno della funzione Lambda.
2. Aggiungi l'annotazione al segmento o al sottosegmento.
3. Termina il segmento o il sottosegmento.

Il seguente esempio di codice mostra come aggiungere un'annotazione a un sottosegmento all'interno di una funzione 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();
}
```

L'X-Ray SDK registra le annotazioni come coppie chiave-valore in un `annotations` oggetto nel documento del segmento. La chiamata all'`addAnnotation`operazione due volte con la stessa chiave sovrascrive un valore registrato in precedenza sullo stesso segmento o sottosegmento.

Per trovare tracciamenti con annotazioni contenenti valori specifici, utilizza la parola chiave `annotation[`*`key`*`]` in un'espressione filtro.

Registrazione di metadati con X-Ray SDK for .NET

Usa i metadati per registrare informazioni su segmenti o sottosegmenti che non devi indicizzare per utilizzarli all'interno di una ricerca. I valori dei metadati possono essere stringhe, numeri, valori booleani o qualsiasi altro oggetto che può essere serializzato in un oggetto o in un array JSON.

Per registrare i metadati

1. Ottieni un'istanza di`AWSXRayRecorder`, come mostrato nel seguente esempio di codice:

   ```
   using Amazon.XRay.Recorder.Core;
   ...
   AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
   ```

2. Chiama `AddMetadata` con uno spazio dei nomi di stringa, una chiave di stringa e un valore di oggetto, come illustrato nel seguente esempio di codice:

   ```
   recorder.AddMetadata("my namespace", "my key", "my value");
   ```

   È inoltre possibile chiamare l'`AddMetadata`operazione utilizzando solo una coppia chiave/valore, come illustrato nel seguente esempio di codice:

   ```
   recorder.AddMetadata("my key", "my value");
   ```

Se non si specifica un valore per lo spazio dei nomi, viene utilizzato l'SDK X-Ray. `default` La chiamata all'`AddMetadata`operazione due volte con la stessa chiave sovrascrive un valore registrato in precedenza sullo stesso segmento o sottosegmento.

Lavorare con Ruby

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Esistono due modi per strumentare l'applicazione Ruby per inviare tracce a X-Ray:

- AWS Distro for OpenTelemetry Ruby: una AWS distribuzione che fornisce un set di librerie open source per l'invio di metriche e tracce correlate a più soluzioni di AWS monitoraggio, tra cui Amazon AWS X-Ray e Amazon OpenSearch Service CloudWatch, tramite Distro for Collector.AWS OpenTelemetry
- AWS X-Ray SDK for Ruby — Un insieme di librerie per la generazione e l'invio di tracce a X-Ray tramite il demone X-Ray.

Per ulteriori informazioni, consulta Scelta tra AWS Distro for OpenTelemetry e X-Ray SDKs.

AWS Distro per Ruby OpenTelemetry

Con AWS Distro for OpenTelemetry (ADOT) Ruby, puoi strumentare le tue applicazioni una sola volta e inviare metriche e tracce correlate a più AWS soluzioni di monitoraggio tra cui Amazon e Amazon Service CloudWatch. AWS X-Ray OpenSearch L'utilizzo di X-Ray con ADOT richiede due componenti: un OpenTelemetry SDK abilitato per l'uso con X-Ray e AWS Distro for OpenTelemetry Collector abilitato per l'uso con X-Ray.

Per iniziare, consulta la documentazione di Distro for Ruby.AWS OpenTelemetry

Per ulteriori informazioni sull'utilizzo di AWS Distro for OpenTelemetry with AWS X-Ray e altro Servizi AWS, consultate Distro for OpenTelemetry o AWS Distro for Documentation.AWS OpenTelemetry

Per ulteriori informazioni sul supporto e l'utilizzo della lingua, vedere AWS Observability on. GitHub

AWS X-Ray SDK per Ruby

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

X-Ray SDK è una libreria per applicazioni web Ruby che fornisce classi e metodi per generare e inviare dati di traccia al demone X-Ray. I dati di traccia includono informazioni sulle richieste HTTP in entrata fornite dall'applicazione e sulle chiamate effettuate dall'applicazione ai servizi downstream utilizzando l' AWS SDK, i client HTTP o un client di registrazione attivo. Puoi anche possibile creare manualmente dei segmenti e aggiungere informazioni di debug in annotazioni e metadati.

Puoi scaricare l'SDK aggiungendolo al tuo gemfile ed eseguendo `bundle install`.

Example Gemfile

```
gem 'aws-sdk'
```

Se usi Rails, inizia aggiungendo il middleware X-Ray SDK per tracciare le richieste in arrivo. Un filtro sulla richiesta crea un segmento. Fino a che il segmento è aperto, puoi usare i metodi del client dell'SDK per aggiungere informazioni al segmento e creare sottosegmenti per tracciare le chiamate a valle. L'SDK, inoltre, registra automaticamente le eccezioni sollevate dall'applicazione per il tempo durante il quale il segmento è aperto. Per applicazioni non Rail, puoi creare i segmenti manualmente.

Successivamente, usa l'SDK X-Ray per strumentare i tuoi AWS SDK per Ruby client HTTP e SQL configurando il registratore per applicare patch alle librerie associate. Ogni volta che effettui una chiamata a un downstream Servizio AWS o a una risorsa con un client dotato di strumenti, l'SDK registra le informazioni sulla chiamata in un sottosegmento. Servizi AWS e le risorse a cui accedi all'interno dei servizi vengono visualizzate come nodi a valle sulla mappa di traccia per aiutarti a identificare errori e problemi di limitazione sulle singole connessioni.

Una volta presa confidenza con l'SDK, personalizza il suo comportamento configurando il registratore. Puoi aggiungere dei plugin per memorizzare i dati sulle risorse di elaborazione sulle

quali è eseguita la tua applicazione, personalizzare il comportamento di campionamento definendo regole di campionatura e impostare un sistema di generazione dei log per visualizzare più o meno informazioni generate dall'SDK nei log dell'applicazione.

Registra ulteriori informazioni sulle richieste e sull'attività svolta dalla tua applicazione in annotazioni e metadati. Le annotazioni sono semplici coppie chiave-valore indicizzati per l'uso con espressioni filtro, in modo da poter cercare tracce che contengono dati specifici. Le immissioni di metadati sono meno restrittive e possono registrare interi oggetti e array, tutto ciò che può essere serializzato in JSON.

> **Annotazioni e metadata**
>
> Le annotazioni e i metadati sono testo arbitrario che aggiungi ai segmenti con X-Ray SDK. Le annotazioni vengono indicizzate per essere utilizzate con le espressioni di filtro. I metadati non sono indicizzati, ma possono essere visualizzati nel segmento non elaborato con la console X-Ray o l'API. Chiunque conceda l'accesso in lettura a X-Ray può visualizzare questi dati.

Quando disponi di una notevole quantità di client analizzati nel tuo codice, un singolo segmento per la richiesta può contenere un numero elevato di sottosegmenti, uno per ciascuna delle chiamate effettuate con un client analizzato. Puoi organizzare e raggruppare i sottosegmenti inglobando le chiamate del client sottosegmenti personalizzati. Puoi creare un sottosegmento personalizzato per un'intera funzione o qualsiasi porzione di codice, e memorizzare metadati e annotazioni sul sottosegmento invece di scrivere tutto all'interno del segmento padre.

Per la documentazione di riferimento per le classi e i metodi dell'SDK, consulta l'AWS X-Ray SDK for Ruby API Reference.

Requisiti

L'X-Ray SDK richiede Ruby 2.3 o successivo ed è compatibile con le seguenti librerie:

- AWS SDK per Ruby versione 3.0 o successiva
- Rails versione 5.1 o successive

Configurazione dell' X-Ray SDK per Ruby

> **Note**
>
> X-Ray SDK/Daemon Avviso di manutenzione: il 25 febbraio 2026, AWS X-Ray SDKs/ Daemon entrerà in modalità di manutenzione, in cui AWS verranno limitate le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, consulta Migrazione da X-Ray strumentazione a strumentazione. OpenTelemetry

L' X-Ray SDK for Ruby ha una `XRay.recorder` classe denominata che fornisce il registratore globale. Puoi configurare il registratore globale per personalizzare il middleware che crea i segmenti relativi alle chiamate HTTP in entrata.

Sections

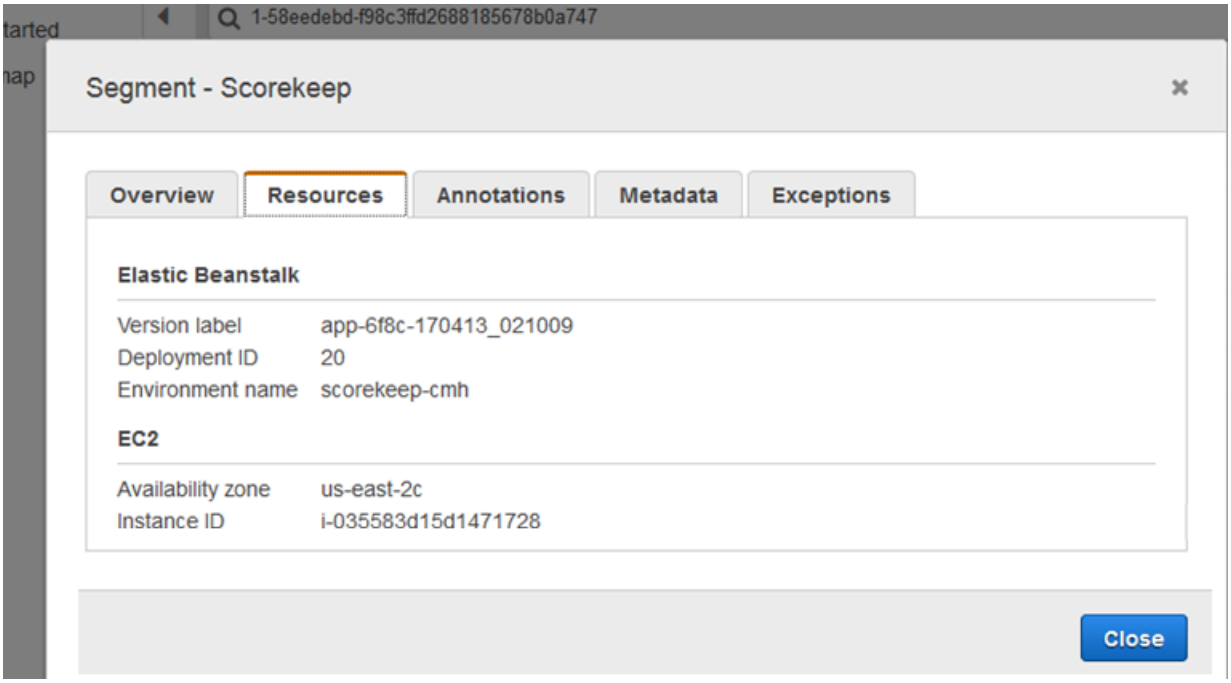
- Plugin di servizio
- Regole di campionamento
- Registrazione dei log
- Configurazione del registratore nel codice
- Configurazione del registratore con Rails
- Variabili di ambiente

Plugin di servizio

`plugins`Utilizzatelo per registrare informazioni sul servizio che ospita l'applicazione.

Plugin

- Amazon EC2: `ec2` aggiunge l'ID dell'istanza e la zona di disponibilità.
- Elastic `elastic_beanstalk` Beanstalk: aggiunge il nome dell'ambiente, l'etichetta della versione e l'ID di distribuzione.
- Amazon ECS: `ecs` aggiunge l'ID del contenitore.



Per utilizzare i plugin, specificalo nell'oggetto di configurazione che passi al registratore.

Example main.rb — Configurazione del plugin

```
my_plugins = %I[ec2 elastic_beanstalk]

config = {
  plugins: my_plugins,
  name: 'my app',
}

XRay.recorder.configure(config)
```

Per configurare il registratore puoi anche utilizzare le variabili di ambiente, che hanno la precedenza sui valori impostati nel codice.

L'SDK utilizza anche le impostazioni del plug-in per impostare il `origin` campo sul segmento. Indica il tipo di AWS risorsa che esegue l'applicazione. Quando utilizzi più plugin, l'SDK utilizza il seguente ordine di risoluzione per determinare l'origine: ElasticBeanstalk > EKS > ECS > EC2.

Regole di campionamento

L'SDK utilizza le regole di campionamento definite nella X-Ray console per determinare quali richieste registrare. La regola predefinita tiene traccia della prima richiesta ogni secondo e del 5% di tutte le richieste aggiuntive su tutti i servizi a cui vengono inviate le tracce. X-Ray Crea regole

aggiuntive nella X-Ray console per personalizzare la quantità di dati registrati per ciascuna delle tue applicazioni.

L'SDK applica le regole personalizzate nell'ordine in cui sono definite. Se una richiesta corrisponde a più regole personalizzate, l'SDK applica solo la prima regola.

> **Note**
>
> Se l'SDK non riesce X-Ray a ottenere le regole di campionamento, torna a una regola locale predefinita che prevede la prima richiesta ogni secondo e il 5% di eventuali richieste aggiuntive per host. Ciò può verificarsi se l'host non dispone dell'autorizzazione per chiamare le API di campionamento o non può connettersi al X-Ray demone, che funge da proxy TCP per le chiamate API effettuate dall'SDK.

Puoi anche configurare l'SDK per caricare le regole di campionamento da un documento JSON. L'SDK può utilizzare le regole locali come backup per i casi in cui il X-Ray campionamento non è disponibile o utilizzare esclusivamente regole locali.

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
  }
}
```

Questo esempio definisce una regola personalizzata e una regola predefinita. La regola personalizzata applica una frequenza di campionamento del cinque percento senza alcun numero

minimo di richieste da tracciare per i percorsi. `/api/move/` La regola predefinita tiene traccia della prima richiesta ogni secondo e del 10% delle richieste aggiuntive.

Lo svantaggio della definizione locale delle regole è che l'obiettivo fisso viene applicato da ciascuna istanza del registratore in modo indipendente, anziché essere gestito dal X-Ray servizio. Man mano che si implementano più host, la tariffa fissa si moltiplica, rendendo più difficile il controllo della quantità di dati registrati.

Per configurare le regole di backup, definire un hash per il documento dell'oggetto di configurazione che si passa al registratore.

Example main.rb — Configurazione delle regole di 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)
```

Per archiviare le regole di campionamento in modo indipendente, definire l'hash in un file separato e richiedere i file per inserirli all'interno della propria applicazione.

Example config/sampling-rules.rb

```
my_sampling_rules =  {
  version: 1,
  default: {
    fixed_target: 1,
    rate: 0.1
  }
}
```

Example main.rb — Regola di campionamento da un file

```
require 'aws-xray-sdk'
require 'config/sampling-rules.rb'

config = {
  sampling_rules: my_sampling_rules,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Per utilizzare solo regole locali, richiedere le regole di campionamento e configurare il `LocalSampler`.

Example main.rb — Campionamento di regole locali

```
require 'aws-xray-sdk'
require 'aws-xray-sdk/sampling/local/sampler'

config = {
  sampler: LocalSampler.new,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Puoi anche configurare il registratore globale per disabilitare il campionamento e analizzare tutte le richieste in entrata.

Example main.rb — Disabilita il campionamento

```
require 'aws-xray-sdk'
config = {
  sampling: false,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Registrazione dei log

Per impostazione predefinita, il registratore genera gli eventi a livello di informazioni su `$stdout`. Puoi personalizzare la generazione dei log definendo un logger nell'oggetto di configurazione da passare al registratore.

Example main.rb — Registrazione

```
require 'aws-xray-sdk'
config = {
  logger: my_logger,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Utilizza i log di debug per identificare i problemi, come ad esempio dei sottosegmenti non chiusi, quando generi dei sottosegmenti manualmente.

Configurazione del registratore nel codice

Impostazioni aggiuntive sono disponibili tramite il metodo `configure` su `XRay.recorder`.

- `context_missing`— Imposta per `LOG_ERROR` evitare di generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.
- `daemon_address`— Imposta l'host e la porta del daemon listener. X-Ray
- `name`— Imposta un nome di servizio che l'SDK utilizza per i segmenti.
- `naming_pattern`— Imposta un modello di nome di dominio per utilizzare la denominazione dinamica.
- `plugins`— Registra informazioni sulle AWS risorse dell'applicazione con i plugin.
- `sampling`— Impostare per `false` disabilitare il campionamento.
- `sampling_rules`— Imposta l'hash contenente le regole di campionamento.

Example main.rb — Disabilita le eccezioni mancanti nel contesto

```
require 'aws-xray-sdk'
config = {
  context_missing: 'LOG_ERROR'
}

XRay.recorder.configure(config)
```

Configurazione del registratore con Rails

Se utilizzi il framework Rails, puoi configurare le opzioni sul registratore globale in un file Ruby sotto `app_root/initializers`. L' X-Ray SDK supporta una chiave di configurazione aggiuntiva da utilizzare con Rails.

- `active_record`— Impostato su per `true` registrare i sottosegmenti per le transazioni del database Active Record.

Configura le impostazioni disponibili in un oggetto di configurazione denominato `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
}
```

Variabili di ambiente

È possibile utilizzare le variabili di ambiente per configurare l' X-Ray SDK for Ruby. L'SDK supporta le seguenti variabili:

- `AWS_XRAY_TRACING_NAME`— Imposta un nome di servizio che l'SDK utilizza per i segmenti. Sostituisce il nome del servizio impostato sulla strategia di denominazione dei segmenti del filtro servlet.
- `AWS_XRAY_DAEMON_ADDRESS`— Imposta l'host e la porta del X-Ray daemon listener. Per impostazione predefinita, l'SDK invia i dati di traccia a. `127.0.0.1:2000` Utilizzate questa variabile se avete configurato il demone per l'ascolto su una porta diversa o se è in esecuzione su un host diverso.
- `AWS_XRAY_CONTEXT_MISSING`— Imposta per `RUNTIME_ERROR` generare eccezioni quando il codice strumentato tenta di registrare dati quando nessun segmento è aperto.

  Valori validi

  - `RUNTIME_ERROR`— Genera un'eccezione di runtime.
  - `LOG_ERROR`— Registra un errore e continua (impostazione predefinita).

- `IGNORE_ERROR`— Ignora l'errore e continua.

  Gli errori relativi a segmenti o sottosegmenti mancanti possono verificarsi quando si tenta di utilizzare un client con strumenti nel codice di avvio che viene eseguito quando non è aperta alcuna richiesta o nel codice che genera un nuovo thread.

Le variabili di ambiente sostituiscono i valori impostati nel codice.

Tracciamento delle richieste in arrivo con il middleware X-Ray SDK for Ruby

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Puoi utilizzare l'SDK X-Ray per tracciare le richieste HTTP in entrata che la tua applicazione serve su un'istanza EC2 in Amazon EC2 o Amazon ECS. AWS Elastic Beanstalk

Se utilizzi Rails, utilizza il middleware Rails per analizzare le richieste HTTP in entrata. Quando aggiungi il middleware all'applicazione e configuri il nome di un segmento, X-Ray SDK for Ruby crea un segmento per ogni richiesta campionata. Tutti i segmenti creati da analisi aggiuntive diventano sottosegmenti del segmento a livello di richiesta che offre informazioni sulla richiesta e sulla risposta HTTP. Queste informazioni includono durata, metodo e conclusione della richiesta HTTP.

Ogni segmento ha un nome che identifica l'applicazione nella mappa del servizio. Il segmento può essere denominato staticamente oppure è possibile configurare l'SDK per denominarlo dinamicamente in base all'intestazione dell'host nella richiesta in entrata. La denominazione dinamica consente di raggruppare le tracce in base al nome di dominio nella richiesta e di applicare un nome predefinito se il nome non corrisponde a uno schema previsto (ad esempio, se l'intestazione dell'host è falsificata).

> **Richieste inoltrate**
>
> Se un sistema di bilanciamento del carico o un altro intermediario inoltra una richiesta all'applicazione, X-Ray prende l'IP del client dall'`X-Forwarded-For`intestazione della richiesta anziché dall'IP di origine nel pacchetto IP. L'IP del client registrato per una richiesta inoltrata può essere falsificato, quindi non dovrebbe essere considerato attendibile.

Quando viene inoltrata una richiesta, l'SDK imposta un campo aggiuntivo nel segmento per indicarlo. Se il segmento contiene il campo `x_forwarded_for` impostato su`true`, l'IP del client è stato preso dall'`X-Forwarded-For`intestazione della richiesta HTTP.

Il middleware crea un segmento per ogni richiesta in entrata con un blocco `http` che contiene le informazioni riportate qui di seguito:

- Metodo HTTP: GET, POST, PUT, DELETE, ecc.
- Indirizzo client: l'indirizzo IP del client che ha inviato la richiesta.
- Codice di risposta: il codice di risposta HTTP per la richiesta completata.
- Tempistica: l'ora di inizio (quando è stata ricevuta la richiesta) e l'ora di fine (quando è stata inviata la risposta).
- Agente utente: il `user-agent` codice della richiesta.
- Lunghezza del contenuto: il `content-length` risultato della risposta.

Utilizzo del middleware Rails

Per usare il middleware, aggiorna il tuo gemfile per includere il railtie necessario.

Example Gemfile - rails

```
gem 'aws-xray-sdk', require: ['aws-xray-sdk/facets/rails/railtie']
```

Per utilizzare il middleware, è inoltre necessario configurare il registratore con un nome che rappresenti l'applicazione nella mappa di traccia.

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {
  name: 'my app'
```

```
}
```

Analisi manuale del codice

Se non utilizzi Rails, crea i segmenti manualmente. Puoi creare un segmento per ogni richiesta in arrivo o creare segmenti attorno a client HTTP o AWS SDK con patch per fornire al registratore il contesto in cui aggiungere sottosegmenti.

```
# 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
```

Configurazione di una strategia di denominazione dei segmenti

AWS X-Ray utilizza un nome di servizio per identificare l'applicazione e distinguerla dalle altre applicazioni, database, risorse esterne APIs e risorse utilizzate dall'applicazione. AWS Quando X-Ray SDK genera segmenti per le richieste in entrata, registra il nome del servizio dell'applicazione nel campo del nome del segmento.

L'X-Ray SDK può denominare i segmenti dopo il nome host nell'intestazione della richiesta HTTP. Tuttavia, questa intestazione può essere falsificata, il che potrebbe causare nodi imprevisti nella mappa dei servizi. Per evitare che l'SDK nomini i segmenti in modo errato a causa di richieste con intestazioni host contraffatte, è necessario specificare un nome predefinito per le richieste in entrata.

Se la tua applicazione soddisfa le richieste per più domini, puoi configurare l'SDK in modo che utilizzi una strategia di denominazione dinamica che rifletta questo aspetto nei nomi dei segmenti. Una strategia di denominazione dinamica consente all'SDK di utilizzare il nome host per le richieste che corrispondono a uno schema previsto e di applicare il nome predefinito alle richieste che non lo fanno.

Ad esempio, potresti avere una singola applicazione che serve le richieste a tre sottodomini:, e. `www.example.com` `api.example.com` `static.example.com` È possibile utilizzare una strategia di denominazione dinamica con lo schema `*.example.com` per identificare i segmenti per ogni sottodominio con un nome diverso, ottenendo tre nodi di servizio sulla mappa dei servizi. Se l'applicazione riceve richieste con un nome host che non corrisponde allo schema, sulla mappa dei servizi verrà visualizzato un quarto nodo con un nome di fallback specificato dall'utente.

Per utilizzare lo stesso nome per tutti i segmenti della richiesta, specifica il nome della tua applicazione quando configuri il registratore, come illustrato nelle sezioni precedenti.

Una strategia di denominazione dinamica definisce un modello al quale devono corrispondere i nomi degli host e un nome di default per l'utilizzo qualora il nome dell'host nella richiesta HTTP non corrisponda al modello. Per denominare segmenti in modo dinamico, specificare un modello di denominazione nel config hash.

Example main.rb — Denominazione dinamica

```
config = {
  naming_pattern: '*mydomain*',
  name: 'my app',
}

XRay.recorder.configure(config)
```

Puoi utilizzare '*' nel modello per una corrispondenza con qualsiasi stringa o '?' per una corrispondenza con qualsiasi carattere singolo.

> **Note**
>
> È possibile sovrascrivere il nome di servizio predefinito definito nel codice con la variabile di ambiente. `AWS_XRAY_TRACING_NAME`

Applicare le patch alle librerie per analizzare le chiamate a valle

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Per strumentare le chiamate downstream, utilizzate l'X-Ray SDK for Ruby per applicare patch alle librerie utilizzate dall'applicazione. L'X-Ray SDK for Ruby può applicare patch alle seguenti librerie.

Librerie supportate

- `net/http`— Client HTTP dello strumento.
- `aws-sdk`— AWS SDK per Ruby Client dello strumento.

Quando si utilizza una libreria con patch, l'X-Ray SDK for Ruby crea un sottosegmento per la chiamata e registra le informazioni dalla richiesta e dalla risposta. Un segmento deve essere disponibile affinché l'SDK possa creare il sottosegmento, sia tramite il middleware dell'SDK che tramite una chiamata a `XRay.recorder.begin_segment`.

Per applicare patch alle librerie, specificatele nell'oggetto di configurazione che passate al registratore X-Ray.

Example main.rb — Librerie di patch

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http aws_sdk]
}

XRay.recorder.configure(config)
```

Tracciamento delle chiamate AWS SDK con X-Ray SDK for Ruby

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Quando l'applicazione effettua chiamate per Servizi AWS archiviare dati, scrivere in una coda o inviare notifiche, X-Ray SDK for Ruby tiene traccia delle chiamate downstream in sottosegmenti. Le risorse tracciate Servizi AWS e a cui accedi all'interno di tali servizi (ad esempio, un bucket Amazon S3 o una coda Amazon SQS) vengono visualizzate come nodi downstream sulla mappa di traccia nella console X-Ray.

L'X-Ray SDK for Ruby strumenta automaticamente AWS tutti i client SDK quando applichi una patch alla libreria. `aws-sdk` Non puoi analizzare singoli client.

Per tutti i servizi, puoi vedere il nome dell'API richiamata nella console X-Ray. Per un sottoinsieme di servizi, l'SDK X-Ray aggiunge informazioni al segmento per fornire una maggiore granularità nella mappa dei servizi.

Ad esempio, quando si effettua una chiamata con un client DynamoDB con strumentazione, l'SDK aggiunge il nome della tabella al segmento per le chiamate destinate a una tabella. Nella console, ogni tabella appare come un nodo separato nella mappa dei servizi, con un nodo DynamoDB generico per le chiamate che non hanno come destinazione una tabella.

Example Sottosegmento per una chiamata a DynamoDB per salvare un elemento

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
```

```
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Quando si accede alle risorse con nome, le chiamate ai seguenti servizi creano ulteriori nodi della mappa del servizio. Le chiamate che non sono hanno come obiettivo risorse specifiche creano un nodo generico per il servizio.

- Amazon DynamoDB: nome della tabella
- Amazon Simple Storage Service: nome del bucket e della chiave
- Amazon Simple Queue Service: nome della coda

Generazione di sottosegmenti personalizzati con X-Ray SDK

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

I sottosegmenti estendono il segmento di una traccia con dettagli sul lavoro svolto per soddisfare una richiesta. Ogni volta che si effettua una chiamata con un client dotato di strumentazione, l'X-Ray SDK registra le informazioni generate in un sottosegmento. È possibile creare sottosegmenti aggiuntivi per raggruppare altri sottosegmenti, misurare le prestazioni di una sezione di codice o registrare annotazioni e metadati.

Per gestire i sottosegmenti, utilizza i metodi `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
```

Per creare un subsegment per una funzione, eseguire il wrapping in una chiamata a `XRay.recorder.capture`.

```
XRay.recorder.capture('name_for_subsegment') do |subsegment|
  resp = myfunc() # myfunc is your function
  subsegment.annotations.update k1: 'v1'
  resp
end
```

Quando create un sottosegmento all'interno di un segmento o di un altro sottosegmento, X-Ray SDK genera un ID per esso e registra l'ora di inizio e l'ora di fine.

Example Sottosegmento con metadati

```
"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"
    }
  },
```

Aggiungi annotazioni e metadati ai segmenti con l'X-Ray SDK for Ruby

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support Ti consigliamo di migrare

a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

È possibile registrare informazioni aggiuntive sulle richieste, sull'ambiente o sull'applicazione con annotazioni e metadati. È possibile aggiungere annotazioni e metadati ai segmenti creati da X-Ray SDK o ai sottosegmenti personalizzati creati dall'utente.

Le annotazioni sono coppie chiave-valore con stringhe, numeri o valori booleani. Le annotazioni sono indicizzate per essere utilizzate con le espressioni di filtro. Utilizzale per registrare i dati che desideri utilizzare per raggruppare le tracce nella console oppure per chiamare l'API `GetTraceSummaries`.

I metadati sono coppie chiave-valore che possono avere valori di qualsiasi tipo, inclusi oggetti ed elenchi, ma non sono indicizzati per essere utilizzati con le espressioni di filtro. Utilizzate i metadati per registrare dati aggiuntivi che desiderate archiviare nella traccia ma che non è necessario utilizzare con la ricerca.

Oltre ad annotazioni e metadati, sui segmenti puoi anche registrare le stringhe degli ID utente. IDs Gli utenti vengono registrati in un campo separato sui segmenti e indicizzati per essere utilizzati nella ricerca.

Sections

- Registrazione delle annotazioni con X-Ray SDK for Ruby
- Registrazione di metadati con X-Ray SDK for Ruby
- Utente che registra IDs con X-Ray SDK for Ruby

Registrazione delle annotazioni con X-Ray SDK for Ruby

Utilizza le annotazioni per memorizzare le informazioni su segmenti o sottosegmenti che desideri siano indicizzate per la ricerca.

Requisiti per le annotazioni

- Chiavi: la chiave per un'annotazione a raggi X può contenere fino a 500 caratteri alfanumerici. Non è possibile utilizzare spazi o simboli diversi da un punto o un punto (.)
- Valori: il valore di un'annotazione X-Ray può contenere fino a 1.000 caratteri Unicode.
- Il numero di annotazioni: è possibile utilizzare fino a 50 annotazioni per traccia.

Per registrare le annotazioni

1. Ottenere un riferimento al segmento o sottosegmento corrente da `xray_recorder`.

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_segment
   ```

   or

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_subsegment
   ```

2. Chiama `update` con un valore hash.

   ```
   my_annotations = { id: 12345 }
   document.annotations.update my_annotations
   ```

   Di seguito è riportato un esempio che mostra come effettuare una chiamata `update` con una chiave di annotazione che contiene un punto.

   ```
   my_annotations = { testkey.test: 12345 }
   document.annotations.update my_annotations
   ```

L'SDK memorizza le annotazioni come coppie chiave-valore in un oggetto `annotations` all'interno del documento di segmento. Se chiami `add_annotations` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Per trovare tracciamenti con annotazioni contenenti valori specifici, utilizza la parola chiave `annotation[`*`key`*`]` in un'espressione filtro.

Registrazione di metadati con X-Ray SDK for Ruby

Utilizza i metadati per memorizzare le informazioni su segmenti o sottosegmenti che non è necessario che siano indicizzate per la ricerca. I valori dei metadati possono essere stringhe, numeri, valori booleani o qualsiasi oggetto che possa essere serializzato in un oggetto o in un vettore JSON.

Per registrare i metadati

1. Ottenere un riferimento al segmento o sottosegmento corrente da `xray_recorder`.

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_segment
   ```

   or

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_subsegment
   ```

2. Chiama `metadata` con una chiave di tipo Stringa, un valore booleano, numerico, di tipo Stringa o di tipo oggetto e un namespace di tipo stringa.

   ```
   my_metadata = {
     my_namespace: {
       key: 'value'
     }
   }
   subsegment.metadata my_metadata
   ```

Se chiami `metadata` due volte con la stessa chiave, il valore precedentemente memorizzato nello stesso segmento o sottosegmento viene sovrascritto.

Utente che registra IDs con X-Ray SDK for Ruby

Registra i segmenti dell'utente IDs su richiesta per identificare l'utente che ha inviato la richiesta.

Per registrare l'utente IDs

1. Ottenere un riferimento al segmento corrente da `xray_recorder`.

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_segment
   ```

2. Imposta il campo utente sul segmento a un ID di tipo stringa dell'utente che ha inviato la richiesta.

```
segment.user = 'U12345'
```

Puoi impostare l'utente nei tuoi controller per registrare l'ID utente non appena l'applicazione inizia ad elaborare una richiesta.

Per trovare tracciamenti associati ad un ID utente, utilizza la parola chiave `user` in un'espressione filtro.

Cronologia di X-Ray SDK e Daemon Support

La tabella seguente elenca le date e il livello di supporto per X-Ray SDKs e Daemon.

Fase SDK e daemon	Data di inizio	Data di fine	Support fornito
Disponibilità generale	N/A	25 febbraio 2026	X-Ray SDKs e Daemon sono completamente supportati. AWS fornisce versioni regolari di SDK e daemon che includono correzioni di bug e sicurezza.
modalità di manutenzione	25 febbraio 2026	N/D	AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Non SDKs/Daemon riceveranno nuovi miglioramenti delle funzionalità.

Ti consigliamo di migrare a OpenTelemetry soluzioni per la strumentazione dell'applicazione e l'invio di tracce a X-Ray. AWS Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla strumentazione a raggi X alla strumentazione. OpenTelemetry

Migrazione dalla strumentazione a raggi X alla strumentazione OpenTelemetry

> **Note**
>
> Avviso di SDK/Daemon manutenzione a raggi X: il 25 febbraio 2026, X-Ray SDKs/Daemon entrerà in modalità di manutenzione, in cui AWS limiterà le versioni di AWS X-Ray SDK e Daemon per risolvere solo problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta. Cronologia di X-Ray SDK e Daemon Support

X-Ray sta passando a OpenTelemetry (OTel) come standard di strumentazione principale per il tracciamento e l'osservabilità delle applicazioni. Questo cambiamento strategico si allinea AWS alle migliori pratiche del settore e offre ai clienti una soluzione più completa, flessibile e pronta per il futuro per le loro esigenze di osservabilità. OpenTelemetryl'ampia adozione nel settore consente il tracciamento delle richieste su diversi sistemi, compresi quelli esterni AWS che potrebbero non integrarsi direttamente con X-Ray.

Questo capitolo fornisce raccomandazioni per una transizione senza intoppi e sottolinea l'importanza della migrazione a soluzioni OpenTelemetry basate su di essa per garantire supporto e accesso continui alle funzionalità più recenti in materia di strumentazione e osservabilità delle applicazioni.

Si consiglia di adottarla OpenTelemetry come soluzione di osservabilità per la strumentazione dell'applicazione.

Argomenti

- Comprensione OpenTelemetry
- Comprensione dei concetti relativi alla migrazione OpenTelemetry
- Panoramica sulla migrazione
- Migrazione da X-Ray Daemon AWS CloudWatch a un agente o un raccoglitore OpenTelemetry
- OpenTelemetry Migrazione a Java
- Esegui la migrazione a OpenTelemetry Go
- Migrare a Node.js OpenTelemetry
- Esegui la migrazione a.NET OpenTelemetry
- Migrare a Python OpenTelemetry

- Migra a Ruby OpenTelemetry

Comprensione OpenTelemetry

OpenTelemetry è un framework di osservabilità standard del settore che fornisce protocolli e strumenti standardizzati per la raccolta di dati di telemetria. Offre un approccio unificato alla strumentazione, alla generazione, alla raccolta e all'esportazione di dati di telemetria come metriche, log e tracce.

Quando si esegue la migrazione da SDKs X-Ray OpenTelemetry a, si ottengono i seguenti vantaggi:

- Supporto avanzato per framework e strumentazione di libreria
- Support per linguaggi di programmazione aggiuntivi
- Funzionalità di strumentazione automatica
- Opzioni flessibili di configurazione del campionamento
- Raccolta unificata di metriche, log e tracce

Il OpenTelemetry raccoglitore offre più opzioni per i formati di raccolta dei dati e le destinazioni di esportazione rispetto al demone X-Ray.

OpenTelemetry supporto in AWS

AWS offre diverse soluzioni per lavorare con OpenTelemetry:

- AWS Distro per OpenTelemetry

  Esporta OpenTelemetry le tracce come segmenti in X-Ray.

  Per ulteriori informazioni, vedete AWS Distro for. OpenTelemetry
- CloudWatch Segnali applicativi

  Esporta OpenTelemetry tracce e metriche personalizzate per monitorare lo stato delle applicazioni.

  Per ulteriori informazioni, consulta Working with Application Signals.
- CloudWatch OTel Endpoint

  Esporta OpenTelemetry le tracce su X-Ray utilizzando l' OTel endpoint HTTP con strumentazione nativa. OpenTelemetry

Per ulteriori informazioni, vedere Utilizzo degli endpoint. OTel

Utilizzo con OpenTelemetry AWS CloudWatch

AWS CloudWatch supporta OpenTelemetry le tracce tramite strumentazione applicativa lato client e AWS CloudWatch servizi nativi come Application Signals, Trace, Map, Metrics e Logs. Per ulteriori informazioni, consulta OpenTelemetry.

Comprensione dei concetti relativi alla migrazione OpenTelemetry

La tabella seguente mappa i concetti relativi ai raggi X ai loro OpenTelemetry equivalenti. La comprensione di queste mappature consente di tradurre la strumentazione a raggi X esistente in: OpenTelemetry

Concetto X-Ray	OpenTelemetry concetto
Registratore a raggi X	Tracer Provider e Tracer
Plugin di servizio	Rilevatore di risorse
Segment	(Server) Span
Sottosegmento	Intervallo (non server)
Regole di campionamento a raggi X	OpenTelemetry Campionamento (personal izzabile)
Emettitore di raggi X	Span Exporter (personalizzabile)
Annotazioni/metadati	Attributes
Strumentazione della biblioteca	Strumentazione per biblioteche
Contesto della traccia a raggi X	Contesto Span
Propagazione del contesto di tracce a raggi X	Propagazione del contesto di traccia W3C
Campionamento di tracce a raggi X	OpenTelemetry Campionamento delle tracce

Concetto X-Ray	OpenTelemetry concetto
N/D	Elaborazione Span
N/D	Bagagli
Demone X-Ray	OpenTelemetry Collezionista

> **Note**
>
> Per ulteriori informazioni sui OpenTelemetry concetti, consulta la OpenTelemetry documentazione.

Confronto delle funzionalità

La tabella seguente mostra quali funzionalità sono supportate in entrambi i servizi. Utilizzate queste informazioni per identificare eventuali lacune da colmare durante la migrazione:

Funzionalità	Strumentazione a raggi X	OpenTelemetry strumentazione
Strumentazione per bibliotec he	Supportata	Supportata
Campionamento a raggi X	Supportata	Supportato in Java/.net/Go OTel Supportato in ADOT Java/. NET/Python/Node.js
Propagazione del contesto di tracce a raggi X	Supportata	Supportata
Rilevamento delle risorse	Supportata	Supportata
Annotazioni sui segmenti	Supportata	Supportata

Funzionalità	Strumentazione a raggi X	OpenTelemetry strumenta zione
Metadati dei segmenti	Supportata	Supportata
Strumentazione automatica a codice zero	Supportato in Java	Supportato in OTel Java/. NET/Python/Node.js Supportato in ADOT Java/. NET/Python/Node.js
Traccia manualmente la creazione	Supportata	Supportata

Impostazione e configurazione del tracciamento

Per creare tracce in OpenTelemetry, è necessario un tracer. Puoi ottenere un tracer inizializzando un Tracer Provider nella tua applicazione. È simile a come si utilizza il registratore a raggi X per configurare X-Ray e creare segmenti e sottosegmenti in una traccia a raggi X.

> **Note**
>
> Il OpenTelemetry Tracer Provider offre più opzioni di configurazione rispetto al registratore X-Ray.

Comprensione della struttura dei dati di traccia

Dopo aver compreso i concetti di base e le mappature delle funzionalità, puoi conoscere dettagli di implementazione specifici come la struttura dei dati di traccia e il campionamento.

OpenTelemetry utilizza gli intervalli anziché segmenti e sottosegmenti per strutturare i dati di traccia. Ogni intervallo include i seguenti componenti:

- Name
- ID univoco
- Timestamp di inizio e fine
- Tipo di intervallo

- Amplia il contesto
- Attributi (metadati chiave-valore)
- Eventi (registri con data e ora)
- Collegamenti ad altri span
- Informazioni sullo stato
- Riferimenti Parent Span

Quando si esegue la migrazione a OpenTelemetry, gli intervalli vengono convertiti automaticamente in segmenti o sottosegmenti X-Ray. Ciò garantisce che l'esperienza esistente con la console rimanga invariata. CloudWatch

Lavorare con gli attributi span

L'SDK X-Ray offre due modi per aggiungere dati a segmenti e sottosegmenti:

Annotazioni

Coppie chiave-valore indicizzate per il filtraggio e la ricerca

Metadati

Coppie chiave-valore contenenti dati complessi che non sono indicizzati per la ricerca

Per impostazione predefinita, gli attributi OpenTelemetry span vengono convertiti in metadati nei dati grezzi X-Ray. Per convertire invece attributi specifici in annotazioni, aggiungete le relative chiavi all'elenco degli attributi. `aws.xray.annotations`

- Per ulteriori informazioni sui OpenTelemetry concetti, vedete OpenTelemetry Traces
- Per dettagli su come i OpenTelemetry dati vengono mappati ai dati a raggi X, vedere sulla traduzione del modello di dati OpenTelemetry a raggi X

Rilevamento delle risorse nell'ambiente

OpenTelemetry utilizza Resource Detectors per raccogliere metadati sulle risorse che generano dati di telemetria. Questi metadati vengono archiviati come attributi delle risorse. Ad esempio, un'entità che produce telemetria potrebbe essere un cluster Amazon ECS o un'istanza Amazon EC2 e gli attributi delle risorse che possono essere registrati da queste entità possono includere l'ARN del cluster Amazon ECS o l'ID dell'istanza Amazon EC2.

- Per informazioni sui tipi di risorse supportati, consulta Resource Semantic Conventions OpenTelemetry
- Per informazioni sui plugin del servizio X-Ray, vedere Configurazione dell'SDK X-Ray

Gestione delle strategie di campionamento

Trace Sampling ti aiuta a gestire i costi raccogliendo dati da un sottoinsieme rappresentativo di richieste anziché da tutte le richieste. OpenTelemetry Sia X-Ray che X-Ray supportano il campionamento, ma lo implementano in modo diverso.

> **Note**
>
> Il campionamento di meno del 100% delle tracce riduce i costi di osservabilità, mantenendo al contempo informazioni significative sulle prestazioni dell'applicazione.

OpenTelemetry offre diverse strategie di campionamento integrate e consente di crearne di personalizzate. Puoi anche configurare un campionatore remoto X-Ray in alcuni linguaggi SDK con cui utilizzare le regole di campionamento X-Ray. OpenTelemetry

Le strategie di campionamento aggiuntive di sono: OpenTelemetry

- Campionamento basato sui genitori: rispetta la decisione di campionamento dell'intervallo principale prima di applicare strategie di campionamento aggiuntive
- Campionamento basato sul rapporto Trace ID — >Campiona in modo casuale una percentuale specificata di intervalli
- Campionamento della coda: applica le regole di campionamento per completare le tracce nel Collector OpenTelemetry
- Campionatori personalizzati: implementate la vostra logica di campionamento utilizzando l'interfaccia di campionamento

Per informazioni sulle regole di campionamento a raggi X, vedere Regole di campionamento nella console X-Ray

Per informazioni sul campionamento della OpenTelemetry coda, vedere Processore di campionamento della coda

Gestione del contesto di traccia

X-Ray SDKs gestisce il contesto del segmento per gestire correttamente le relazioni padre-figlio tra segmenti e sottosegmenti in una traccia. OpenTelemetry utilizza un meccanismo simile per garantire che gli span abbiano l'intervallo principale corretto. Memorizza e propaga i dati di tracciamento in un contesto di richiesta. Ad esempio, quando l'applicazione elabora una richiesta e crea uno span del server per rappresentare tale richiesta, OpenTelemetry memorizzerà l'estensione del server nel OpenTelemetry Context in modo che, quando viene creato uno span secondario, tale span figlio possa fare riferimento allo span nel Context come riferimento principale.

Propagazione del contesto di traccia

Sia X-Ray che OpenTelemetry utilizzano le intestazioni HTTP per propagare il contesto di traccia tra i servizi. Ciò consente di collegare i dati di traccia generati da diversi servizi e di mantenere le decisioni di campionamento.

L'SDK X-Ray propaga automaticamente il contesto della traccia utilizzando l'intestazione di traccia X-Ray. Quando un servizio ne chiama un altro, l'intestazione di traccia contiene il contesto necessario per mantenere le relazioni padre-figlio tra le tracce.

OpenTelemetry supporta diversi formati di trace header per la propagazione del contesto, tra cui:

- W3C Trace Context (impostazione predefinita)
- Intestazione di tracciamento X-Ray
- Altri formati personalizzati

> **Note**
>
> È possibile configurare l'utilizzo OpenTelemetry di uno o più formati di intestazione. Ad esempio, utilizzate X-Ray Propagator per inviare il contesto di traccia ai AWS servizi che supportano il tracciamento a raggi X.

Configura e usa X-Ray Propagator per abilitare il tracciamento tra i servizi. AWS Ciò consente di propagare il contesto di traccia agli endpoint API Gateway e ad altri servizi che supportano X-Ray.

- Per informazioni sugli X-Ray Trace Header, vedere Tracing header nella X-Ray Developer Guide

- Per informazioni sulla propagazione del OpenTelemetry contesto, vedere Context and Context Propagation nella documentazione OpenTelemetry

Utilizzo della strumentazione della libreria

Sia X-Ray che OpenTelemetry forniscono una strumentazione di libreria che richiede modifiche minime al codice per aggiungere tracciamento alle applicazioni.

X-Ray fornisce funzionalità di strumentazione di libreria. Ciò consente di aggiungere strumentazioni X-Ray predefinite con modifiche minime al codice dell'applicazione. Queste strumentazioni supportano librerie specifiche come i client AWS SDK e HTTP, nonché framework web come Spring Boot o Express.js.

OpenTelemetryle librerie di strumentazione generano intervalli dettagliati per le librerie tramite hook di libreria o modifiche automatiche del codice, che richiedono modifiche minime al codice.

Per determinare se OpenTelemetry Library Instrumentations supporta la tua libreria, cercala nel Registro di sistema. OpenTelemetry OpenTelemetry

Esportazione di tracce

X-Ray e OpenTelemetry utilizza diversi metodi per esportare i dati di traccia.

Esportazione di tracce a raggi X

Gli X-Ray SDKs utilizzano un emettitore per inviare dati di traccia:

- Invia segmenti e sottosegmenti al demone X-Ray
- Utilizza UDP per I/O non bloccanti
- Configurato per impostazione predefinita nell'SDK

OpenTelemetry traccia l'esportazione

OpenTelemetry utilizza Span Exporters configurabili per inviare dati di traccia:

- Utilizza i protocolli http/protobuf o grpc
- Esporta gli intervalli verso gli endpoint monitorati dal Collector o dall'Agent OpenTelemetry CloudWatch
- Consente configurazioni di esportazione personalizzate

Elaborazione e inoltro delle tracce

X-Ray e OpenTelemetry forniscono componenti per ricevere, elaborare e inoltrare dati di traccia.

Elaborazione di tracce a raggi X

Il demone X-Ray gestisce l'elaborazione delle tracce:

- Ascolta il traffico UDP proveniente da X-Ray SDKs
- Esegue in batch segmenti e sottosegmenti
- Carica i batch sul servizio X-Ray

OpenTelemetry elaborazione delle tracce

Il OpenTelemetry Collector gestisce l'elaborazione delle tracce:

- Riceve tracce dai servizi strumentati
- Elabora e modifica facoltativamente i dati di traccia
- Invia tracce elaborate a vari backend, incluso X-Ray

> **Note**
>
> L' AWS CloudWatch agente può anche ricevere e inviare OpenTelemetry tracce a X-Ray. Per ulteriori informazioni, consulta Raccogli metriche e tracce con. OpenTelemetry

Elaborazione Span (concetto specifico) OpenTelemetry

OpenTelemetry utilizza Span Processors per modificare gli span man mano che vengono creati:

- Consente di leggere e modificare gli intervalli al momento della creazione o del completamento
- Abilita una logica personalizzata per la gestione degli intervalli

Bagaglio (concetto specifico) OpenTelemetry

OpenTelemetryla funzione Baggage consente la propagazione di dati chiave-valore:

- Consente il passaggio di dati arbitrari insieme al contesto di traccia
- Utile per diffondere informazioni specifiche dell'applicazione oltre i confini del servizio

Per informazioni sul Collector, vedere OpenTelemetry Collector OpenTelemetry

Per informazioni sui concetti di X-Ray, vedere X-Ray concepts nella X-Ray Developer Guide

Panoramica sulla migrazione

Questa sezione fornisce una panoramica delle modifiche al codice necessarie per la migrazione. L'elenco seguente contiene linee guida specifiche per la lingua e le fasi di migrazione di X-Ray Daemon.

> **Important**
>
> Per migrare completamente dalla strumentazione a raggi X alla strumentazione, OpenTelemetry è necessario:
>
> 1. Sostituisci l'utilizzo di X-Ray SDK con una soluzione OpenTelemetry
> 2. Sostituisci il demone X-Ray con l' CloudWatch agente o il OpenTelemetry raccoglitore (con X-Ray Exporter)

- OpenTelemetry Migrazione a Java
- Esegui la migrazione a OpenTelemetry Go
- Migrare a Node.js OpenTelemetry
- Esegui la migrazione a.NET OpenTelemetry
- Migrare a Python OpenTelemetry
- Migra a Ruby OpenTelemetry

Raccomandazioni per applicazioni nuove ed esistenti

Per le applicazioni nuove ed esistenti, si consiglia di utilizzare le seguenti soluzioni per abilitare la tracciabilità nelle applicazioni:

Instrumentation

- OpenTelemetry SDKs
- AWS Distro per strumentazione OpenTelemetry

Raccolta dati

- OpenTelemetry Collezionista
- CloudWatch Agente

Dopo la migrazione a soluzioni OpenTelemetry basate, la tua CloudWatch esperienza rimarrà la stessa. Potrai comunque visualizzare le tue tracce nello stesso formato nelle pagine Traces e Trace Map della CloudWatch console o recuperare i dati di traccia tramite X-Ray APIs.

Tracciamento delle modifiche alla configurazione

È necessario sostituire la configurazione X-Ray con una OpenTelemetry configurazione.

Confronto tra X-Ray e configurazione OpenTelemetry

Funzionalità	SDK X-Ray	OpenTelemetry
Configurazioni predefinite	• Campionamento centralizzato a raggi X • Propagazione del contesto di X-Ray Trace • Esportazione delle tracce su X-Ray Daemon	• Esportazione di tracce in OpenTelemetry Collector o CloudWatch Agent (HTTP/gRPC) • Propagazione W3C Trace Context
Configurazioni manuali	• Regole di campionamento locali • Plug-in per il rilevamento delle risorse	• Campionamento a raggi X (potrebbe non essere disponibile per tutte le lingue) • Rilevamento delle risorse • Propagazione del contesto di X-Ray Trace

Modifiche alla strumentazione della libreria

Aggiorna il codice per utilizzare OpenTelemetry Library Instrumentation anziché X-Ray Library Instrumentation per AWS SDK, client HTTP, Web Frameworks e altre librerie. Questo genera OpenTelemetry Tracce anziché Tracce X-Ray.

> **Note**
>
> Le modifiche al codice variano in base alla lingua e alla libreria. Per istruzioni dettagliate, consulta le guide alla migrazione specifiche per lingua.

Modifiche alla strumentazione dell'ambiente Lambda

Per utilizzarle OpenTelemetry nelle funzioni Lambda, scegli una di queste opzioni di configurazione:

1. Usa un Lambda Layer con strumentazione automatica:
   - (Consigliato) AWS Lambda Layer per OpenTelemetry

     > **Note**
     >
     > Per utilizzare solo la traccia, imposta la variabile di ambiente Lambda. `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`

   - AWS Lambda Layer gestito per ADOT
2. Configurazione manuale della OpenTelemetry funzione Lambda:
   - Configurazione di un processore Simple Span con un X-Ray UDP Span Exporter
   - Configurare un propagatore Lambda a raggi X

Creazione manuale di dati di traccia

Sostituisci i segmenti e i sottosegmenti X-Ray con Spans: OpenTelemetry

- Usa un Tracer per creare Span OpenTelemetry
- Aggiungi attributi a Spans (equivalente ai metadati e alle annotazioni X-Ray)

> **Important**
>
> Quando viene inviato a X-Ray:
>
> - Gli intervalli del server vengono convertiti in segmenti X-Ray
> - Gli altri intervalli vengono convertiti in sottosegmenti X-Ray
> - Gli attributi vengono convertiti in metadati per impostazione predefinita

Per convertire un attributo in un'annotazione, aggiungete la relativa chiave all'elenco degli `aws.xray.annotations` attributi. Per ulteriori informazioni, vedere Abilitare le annotazioni X-Ray personalizzate.

Migrazione da X-Ray Daemon AWS CloudWatch a un agente o un raccoglitore OpenTelemetry

È possibile utilizzare l' CloudWatch agente o il OpenTelemetry raccoglitore per ricevere tracce dalle applicazioni strumentate e inviarle a X-Ray.

> **Note**
>
> La versione dell' CloudWatch agente 1.300025.0 e successive può raccogliere tracce. OpenTelemetry L'utilizzo dell' CloudWatch agente anziché del demone X-Ray riduce il numero di agenti da gestire. Per ulteriori informazioni, vedere Raccolta di metriche, log e tracce con l'agente. CloudWatch

Sections

- Migrazione su Amazon EC2 o server locali
- Migrazione su Amazon ECS
- Migrazione su Elastic Beanstalk

Migrazione su Amazon EC2 o server locali

> **Important**
>
> Arresta il processo X-Ray Daemon prima di utilizzare l' CloudWatch agente o il OpenTelemetry raccoglitore per evitare conflitti tra le porte.

Configurazione X-Ray Daemon esistente

Installazione del demone

L'utilizzo esistente di X-Ray Daemon è stato installato utilizzando uno di questi metodi:

Installazione manuale

Scarica ed esegui il file eseguibile dal daemon X-Ray Amazon S3.

Installazione automatica

Utilizzate questo script per installare il demone all'avvio di un'istanza:

```
#!/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
```

Configurazione del daemon

L'utilizzo esistente di X-Ray Daemon è stato configurato utilizzando uno dei seguenti metodi:

- Argomenti della riga di comando
- File di configurazione (`xray-daemon.yaml`)

Example Utilizzo di un file di configurazione

```
./xray -c ~/xray-daemon.yaml
```

Esecuzione del daemon

L'utilizzo esistente di X-Ray Daemon è stato avviato con il seguente comando:

```
~/xray-daemon$ ./xray -o -n us-east-1
```

Rimozione del demone

Per rimuovere il demone X-Ray dalla tua istanza Amazon EC2:

1. Arresta il servizio daemon:

   ```
   systemctl stop xray
   ```

2. Eliminare il file di configurazione:

   ```
   rm ~/path/to/xray-daemon.yaml
   ```

3. Se configurato, rimuovi il file di registro:

   > **Note**
   >
   > La posizione del file di registro dipende dalla configurazione:
   >
   > - Configurazione della riga di comando: `/var/log/xray-daemon.log`
   > - File di configurazione: verifica l'`LogPath`impostazione

Configurazione dell' CloudWatch agente

Installazione dell'agente

Per le istruzioni di installazione, vedere Installazione dell' CloudWatch agente su un server locale.

Configurazione dell'agente

1. Crea un file di configurazione per abilitare la raccolta delle tracce. Per ulteriori informazioni, vedere Creazione del file di configurazione CloudWatch dell'agente.
2. Configura le autorizzazioni IAM:
   - Allega un ruolo IAM o specifica le credenziali per l'agente. Per ulteriori informazioni, consulta Configurazione dei ruoli IAM.

- Assicurati che il ruolo o le credenziali includano l'`xray:PutTraceSegments`autorizzazione.

Avvio dell'agente di

Per istruzioni su come avviare l'agente, consulta Avvio dell' CloudWatch agente tramite la riga di comando.

Configurazione del OpenTelemetry raccoglitore

Installazione del collettore

Scaricate e installate il OpenTelemetry raccoglitore per il vostro sistema operativo. Per istruzioni, consultate Installazione del raccoglitore.

Configurazione del raccoglitore

Configura i seguenti componenti nel tuo raccoglitore:

- estensione awsproxy

  Necessario per il campionamento a raggi X
- OTel ricevitori

  Raccoglie le tracce dall'applicazione
- esportatore di raggi x

  Invia tracce a X-Ray

Example Esempio di configurazione del collettore: .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

Configura le AWS credenziali con l'autorizzazione. `xray:PutTraceSegments` Per ulteriori informazioni, vedere Specificazione delle credenziali.

Avvio del raccoglitore

Esegui il raccoglitore con il tuo file di configurazione:

```
otelcol --config=otel-collector-config.yaml
```

Migrazione su Amazon ECS

Important

Il tuo ruolo di attività deve avere l'`xray:PutTraceSegments`autorizzazione per qualsiasi raccoglitore che utilizzi.
Arresta qualsiasi contenitore X-Ray Daemon esistente prima di eseguire l' CloudWatch agente o il contenitore di OpenTelemetry raccolta sullo stesso host per evitare conflitti tra porte.

Utilizzo dell'agente CloudWatch

1. Scarica l'immagine Docker da Amazon ECR Public Gallery.
2. Crea un file di configurazione denominato: `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. Memorizza la configurazione in Systems Manager Parameter Store:
   1. Aprire https://console.aws.amazon.com/systems-manager/
   2. Scegliete Crea parametro
   3. Immetti uno dei seguenti valori:
      - Valore: `/ecs/cwagent/otel-config`
      - Livello: Standard
      - Tipo: String
      - Tipo di dati: testo
      - Valore: [Incolla qui la configurazione cw-agent-otel .json]
4. Crea una definizione di attività utilizzando la modalità di rete bridge:

   Nella tua definizione di attività, la configurazione dipende dalla modalità di rete utilizzata. La modalità di rete bridge è l'impostazione predefinita e può essere utilizzato nel VPC di default. In una rete bridge, imposta la variabile di `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` ambiente per indicare all' OpenTelemetry SDK quali sono l'endpoint e la porta per l' CloudWatch agente.

È inoltre necessario creare un collegamento dal contenitore dell'applicazione al contenitore Collector per inviare le tracce dall' OpenTelemetry SDK dell'applicazione al contenitore Collector.

Example CloudWatch definizione del compito dell'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"
                }
```

```
            ]
        }
    ]
}
```

Per ulteriori informazioni, consulta Implementazione dell' CloudWatch agente per raccogliere i parametri a livello di istanza di Amazon EC2 su Amazon ECS.

OpenTelemetry Utilizzo del raccoglitore

1. Ottieni l'immagine Docker `otel/opentelemetry-collector-contrib` da Docker Hub.
2. Crea un file di configurazione chiamato `otel-collector-config.yaml` utilizzando lo stesso contenuto mostrato nella sezione Amazon EC2 configuring the collector, ma aggiorna gli endpoint per utilizzarli al posto di. `0.0.0.0` `127.0.0.1`
3. Per utilizzare questa configurazione in Amazon ECS, puoi archiviare la configurazione in Systems Manager Parameter Store. Innanzitutto, vai alla console di Systems Manager Parameter Store e scegli Crea nuovo parametro. Crea un nuovo parametro con le seguenti informazioni:
   - Nome:/ecs/otel/config(a questo nome verrà fatto riferimento nella definizione dell'attività per il raccoglitore)
   - Livello: Standard
   - Tipo: String
   - Tipo di dati: testo
   - Valore: [Incolla qui la otel-collector-config configurazione .yaml]
4. Crea una definizione di attività per distribuire il OpenTelemetry collettore utilizzando la modalità di rete bridge come esempio.

   Nella definizione dell'attività, la configurazione dipende dalla modalità di rete utilizzata. La modalità di rete bridge è l'impostazione predefinita e può essere utilizzato nel VPC di default. In una rete bridge, impostate la variabile di `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` ambiente per indicare all' OpenTelemetry SDK quali sono l'endpoint e la porta per il OpenTelemetry Collector. È inoltre necessario creare un collegamento dal contenitore dell'applicazione al contenitore Collector per inviare le tracce dall' OpenTelemetry SDK dell'applicazione al contenitore Collector.

   Example OpenTelemetry definizione del compito del collettore

```
{
```

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

Migrazione su Elastic Beanstalk

> **Important**
>
> Arresta il processo X-Ray Daemon prima di utilizzare l' CloudWatch agente per evitare conflitti tra le porte.

L'integrazione X-Ray Daemon esistente è stata attivata utilizzando la console Elastic Beanstalk o configurando X-Ray Daemon nel codice sorgente dell'applicazione con un file di configurazione.

Utilizzo dell'agente CloudWatch

Sulla piattaforma Amazon Linux 2, configura l' CloudWatch agente utilizzando un file `.ebextensions` di configurazione:

1. Crea una directory denominata `.ebextensions` nella cartella principale del progetto
2. Crea un file denominato `cloudwatch.config` all'interno della `.ebextensions` directory con il seguente contenuto:

```
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. Includi la `.ebextensions` directory nel bundle dei sorgenti dell'applicazione durante la distribuzione

Per ulteriori informazioni sui file di configurazione di Elastic Beanstalk, consulta Personalizzazione avanzata dell'ambiente con file di configurazione.

OpenTelemetry Migrazione a Java

Questa sezione fornisce indicazioni sulla migrazione dall'SDK X-Ray all'SDK per le applicazioni SDK OpenTelemetry for Java.

Sections

- Soluzione di strumentazione automatica a codice zero
- Soluzioni di strumentazione manuale con SDK
- Tracciamento delle richieste in arrivo (strumentazione Spring Framework)
- AWS Strumentazione SDK v2
- Analisi delle chiamate HTTP in uscita
- Supporto strumentale per altre librerie
- Creazione manuale dei dati di traccia
- Strumentazione Lambda

Soluzione di strumentazione automatica a codice zero

With X-Ray Java agent

Per abilitare l'agente X-Ray Java, era necessario modificare gli argomenti JVM dell'applicazione.

```
-javaagent:/path-to-disco/disco-java-agent.jar=pluginPath=/path-to-disco/disco-plugins
```

With OpenTelemetry-based Java agent

Per utilizzare agenti Java OpenTelemetry basati.

- Utilizzate l'agente Java AWS Distro for OpenTelemetry (ADOT) Auto-Instrumentation per la strumentazione automatica con l'agente Java ADOT. Per ulteriori informazioni, vedete

Auto-Instrumentation for Traces and Metrics with the Java agent. Se desideri solo tracciare, disabilita la variabile di `OTEL_METRICS_EXPORTER=none` ambiente per esportare le metriche dall'agente Java.

(Facoltativo) È inoltre possibile abilitare CloudWatch Application Signals durante la strumentazione automatica delle applicazioni AWS con la strumentazione automatica ADOT Java per monitorare lo stato attuale delle applicazioni e tenere traccia delle prestazioni delle applicazioni a lungo termine. Application Signals offre una visione unificata e incentrata sulle applicazioni delle applicazioni, dei servizi e delle dipendenze e aiuta a monitorare e valutare lo stato delle applicazioni. Per ulteriori informazioni, consulta Application Signals.

- Utilizzate l'agente Java per la strumentazione automatica. OpenTelemetry Per ulteriori informazioni, vedete Strumentazione a codice zero con Java Agent.

Soluzioni di strumentazione manuale con SDK

Tracing setup with X-Ray SDK

Per strumentare il codice con X-Ray SDK for Java, prima era necessario configurare la `AWSXRay` classe con plug-in di servizio e regole di campionamento locali, quindi è stato utilizzato un registratore fornito.

```
static { AWS XRayRecorderBuilder builder = AWS
 XRayRecorderBuilder.standard().withPlugin(new EC2Plugin()).withPlugin(new
 ECSPlugin()); AWS XRay.setGlobalRecorder(builder.build());
}
```

Tracing setup with OpenTelemetry SDK

Sono richieste le seguenti dipendenze.

```
<dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>io.opentelemetry</groupId>
                <artifactId>opentelemetry-bom</artifactId>
                <version>1.49.0</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
```

```
                <groupId>io.opentelemetry.instrumentation</groupId>
                <artifactId>opentelemetry-instrumentation-bom</artifactId>
                <version>2.15.0</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>io.opentelemetry</groupId>
            <artifactId>opentelemetry-sdk</artifactId>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry</groupId>
            <artifactId>opentelemetry-api</artifactId>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.semconv</groupId>
            <artifactId>opentelemetry-semconv</artifactId>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry</groupId>
            <artifactId>opentelemetry-exporter-otlp</artifactId>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.contrib</groupId>
            <artifactId>opentelemetry-aws-xray</artifactId>
            <version>1.46.0</version>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.contrib</groupId>
            <artifactId>opentelemetry-aws-xray-propagator</artifactId>
            <version>1.46.0-alpha</version>
        </dependency>
        <dependency>
            <groupId>io.opentelemetry.contrib</groupId>
            <artifactId>opentelemetry-aws-resources</artifactId>
            <version>1.46.0-alpha</version>
        </dependency>
    </dependencies>
```

Configura l' OpenTelemetry SDK creando un'istanza `TracerProvider` e registrando globalmente un oggetto. `OpenTelemetrySdk` Configura questi componenti:

- Un OTLP Span Exporter (ad esempio, OtlpGrpcSpanExporter): necessario per esportare le tracce verso l'agente o il raccoglitore CloudWatch OpenTelemetry
- Un propagatore AWS a raggi X: necessario per propagare il Trace Context ai AWS servizi integrati con X-Ray
- Un campionatore remoto AWS a raggi X: necessario se è necessario campionare le richieste utilizzando le regole di campionamento a raggi X
- Resource Detectors (ad esempio, EcsResource o Ec2Resource): rilevano i metadati dell'host che esegue l'applicazione

```
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();
```

Tracciamento delle richieste in arrivo (strumentazione Spring Framework)

With X-Ray SDK

Per informazioni su come utilizzare X-Ray SDK con il framework Spring per strumentare l'applicazione, consulta AOP with Spring e X-Ray SDK for Java. Per abilitare AOP in Spring, completa questi passaggi.

1. Configurare Spring
2. Aggiungere un filtro di tracciamento all'applicazione
3. Annota il codice o implementa un'interfaccia
4. Attivare X-Ray nell'applicazione

With OpenTelemetry SDK

OpenTelemetry fornisce librerie di strumentazione per raccogliere tracce per le richieste in arrivo per le applicazioni Spring Boot. Per abilitare la strumentazione Spring Boot con una configurazione minima, includi la seguente dipendenza.

```
<dependency>
           <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-spring-boot-starter</artifactId>
        </dependency>
```

Per ulteriori informazioni su come abilitare e configurare la strumentazione Spring Boot per la OpenTelemetry configurazione, vedere Guida introduttiva. OpenTelemetry

Using OpenTelemetry-based Java agents

Il metodo predefinito consigliato per strumentare le applicazioni Spring Boot consiste nell'utilizzare l'agente OpenTelemetry Java con strumentazione bytecode, che fornisce anche più out-of-the-box strumentazioni e configurazioni rispetto all'utilizzo diretto dell'SDK. Soluzione di strumentazione automatica a codice zeroPer iniziare, consulta.

AWS Strumentazione SDK v2

With X-Ray SDK

L'X-Ray SDK for Java è in grado di strumentare automaticamente tutti i client AWS SDK v2 quando aggiungi il sottomodulo nella build. `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor`

Per strumentare le chiamate dei client a valle dei singoli client ai AWS servizi con AWS SDK for Java 2.2 e versioni successive, `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` il modulo dalla configurazione di build è stato escluso e `aws-xray-recorder-sdk-aws-sdk-v2` il modulo è stato incluso. I singoli client sono stati strumentati configurandoli con un. `TracingInterceptor`

```
import com.amazonaws.xray.interceptors.TracingInterceptor;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
//...

public class MyModel {
  private DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_WEST_2)
    .overrideConfiguration(
      ClientOverrideConfiguration.builder()
        .addExecutionInterceptor(new TracingInterceptor())
        .build()
      )
    .build();
//...
```

With OpenTelemetry SDK

Per strumentare automaticamente tutti i client AWS SDK, aggiungi il sottomodulo. `opentelemetry-aws-sdk-2.2-autoconfigure`

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-aws-sdk-2.2-autoconfigure</artifactId>
            <version>2.15.0-alpha</version>
            <scope>runtime</scope>
        </dependency>
```

Per strumentare i singoli client AWS SDK, aggiungi il sottomodulo. `opentelemetry-aws-sdk-2.2`

```
<dependency>
            <groupId>io.opentelemetry.instrumentation</groupId>
            <artifactId>opentelemetry-aws-sdk-2.2</artifactId>
            <version>2.15.0-alpha</version>
            <scope>compile</scope>
        </dependency>
```

Quindi, registra un interceptor durante la creazione di un client SDK. AWS

```
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();
```

Analisi delle chiamate HTTP in uscita

With X-Ray SDK

Per strumentare le richieste HTTP in uscita con X-Ray, era necessario l'X-Ray SDK per la versione di Apache di Java. HttpClient

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
  public String randomName() throws IOException {
```

```
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
```

With OpenTelemetry SDK

Analogamente a X-Ray Java SDK, OpenTelemetry fornisce una `ApacheHttpClientTelemetry` classe con un metodo builder che consente la creazione di un'istanza di un set `HttpClientBuilder` per fornire intervalli OpenTelemetry basati e propagazione del contesto per 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>
```

Di seguito è riportato un esempio di codice tratto da. opentelemetry-java-instrumentation Il client HTTP fornito da newHttpClient () genererà tracce per le richieste eseguite.

```
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();
  }
}
```

Supporto strumentale per altre librerie

Trova l'elenco completo delle strumentazioni di libreria supportate per OpenTelemetry Java nel rispettivo GitHub repository di strumentazione, in Librerie, framework, server di applicazioni supportati e. JVMs

In alternativa, puoi cercare nel OpenTelemetry Registro per scoprire se supporta la strumentazione. OpenTelemetry Per iniziare la ricerca, vedi Registro.

Creazione manuale dei dati di traccia

With X-Ray SDK

Con X-Ray SDK, sono necessari i `beginSubsegment` metodi `beginSegment` and per creare manualmente segmenti e sottosegmenti 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

È possibile utilizzare intervalli personalizzati per monitorare le prestazioni delle attività interne che non vengono acquisite dalle librerie di strumentazione. Nota che solo i server di tipo span

vengono convertiti in segmenti X-Ray, tutti gli altri span vengono convertiti in sottosegmenti X-Ray.

Innanzitutto, è necessario creare un Tracer per generare intervalli, che è possibile ottenere tramite il metodo. `openTelemetry.getTracer` Ciò fornirà un'istanza Tracer tra quelle registrate globalmente nell'esempio. `TracerProvider` Soluzioni di strumentazione manuale con SDK È possibile creare tutte le istanze Tracer necessarie, ma è comune avere una sola istanza Tracer per un'intera applicazione.

```
Tracer tracer = openTelemetry.getTracer("my-app");
```

È possibile utilizzare Tracer per creare intervalli.

```
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();
```

Gli intervalli hanno un tipo predefinito di 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();
}
```

Aggiungere annotazioni e metadati alle tracce con SDK OpenTelemetry

Nell'esempio precedente, il `setAttribute` metodo viene utilizzato per aggiungere attributi a ogni intervallo. Per impostazione predefinita, tutti gli attributi span verranno convertiti in metadati nei dati grezzi X-Ray. Per garantire che un attributo venga convertito in un'annotazione e non in metadati, l'esempio precedente aggiunge la chiave di quell'attributo all'elenco dell'attributo. `aws.xray.annotations` Per ulteriori informazioni, consulta Abilitare le annotazioni e le annotazioni a raggi X personalizzate e i metadati.

Con agenti Java basati OpenTelemetry

Se si utilizza l'agente Java per strumentare automaticamente l'applicazione, è necessario eseguire la strumentazione manuale nell'applicazione. Ad esempio, al codice dello strumento all'interno dell'applicazione per sezioni che non sono coperte da alcuna libreria di strumentazione automatica.

Per eseguire la strumentazione manuale con l'agente, è necessario utilizzare l'artefatto. `opentelemetry-api` La versione dell'artefatto non può essere più recente della versione dell'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();
```

Strumentazione Lambda

With X-Ray SDK

Utilizzando l'SDK X-Ray, dopo che Lambda ha abilitato l'Active Tracing, non è richiesta alcuna configurazione aggiuntiva per utilizzare l'SDK X-Ray. Lambda creerà un segmento che rappresenta l'invocazione del gestore Lambda e potrai creare sottosegmenti o librerie di strumenti utilizzando l'SDK X-Ray senza alcuna configurazione aggiuntiva.

With OpenTelemetry-based solutions

Strumentazione automatica Lambda: puoi strumentare automaticamente la tua Lambda AWS con strati Lambda venduti utilizzando le seguenti soluzioni:

- AWS Lambda Layer per OpenTelemetry (consigliato)

> **Note**
>
> Questo livello Lambda dispone di CloudWatch Application Signals abilitati per impostazione predefinita, il che consente il monitoraggio delle prestazioni e dello stato dell'applicazione Lambda raccogliendo sia metriche che tracce.
> Per il semplice tracciamento, imposta la variabile di ambiente Lambda.
> `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`

  - Consente il monitoraggio delle prestazioni e dello stato dell'applicazione Lambda
  - Per impostazione predefinita, raccoglie sia le metriche che le tracce
- AWS layer Lambda gestito per ADOT Java. Per ulteriori informazioni, consulta AWS Distro for OpenTelemetry Lambda Support For Java.

Per utilizzare la strumentazione manuale insieme al livello di strumentazione automatica, vedere. Soluzioni di strumentazione manuale con SDK Per ridurre gli avviamenti a freddo, prendi in considerazione OpenTelemetry l'utilizzo di strumentazione manuale per generare OpenTelemetry tracce per la tua funzione Lambda.

OpenTelemetry strumentazione manuale per Lambda AWS

Considera il seguente codice di funzione Lambda che effettua una chiamata 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);
        }
    }
}
```

Ecco le dipendenze.

```
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')
}
```

Per strumentare manualmente il tuo gestore Lambda e il client Amazon S3, procedi come segue.

1. Sostituisci le classi di funzioni che implementano `RequestHandler` (o RequestStreamHandler) con quelle che estendono `TracingRequestHandler` (o). TracingRequestStreamHandler
2. Crea un'istanza TracerProvider e registra globalmente un OpenTelemetrySdk oggetto. Si TracerProvider consiglia di configurarlo con:
   a. Un Simple Span Processor con un X-Ray UDP Span Exporter per inviare tracce all'endpoint UDP X-Ray di Lambda
   b. Un campionatore sempre attivo (predefinito se non configurato ParentBased)
   c. Una risorsa con service.name impostato sul nome della funzione Lambda
   d. Un propagatore Lambda a raggi X
3. Modificate il `handleRequest` metodo `doHandleRequest` e passate l'`OpenTelemetrySdk`oggetto alla classe base.
4. Strumenta il client Amazon S3 con la strumentazione OpenTemetry AWS SDK registrando l'interceptor durante la creazione del client.

Sono OpenTelemetry necessarie le seguenti dipendenze correlate.

```
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')
}
```

Il codice seguente illustra la funzione Lambda dopo le modifiche richieste. È possibile creare intervalli personalizzati aggiuntivi per completare gli intervalli forniti 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));
    }

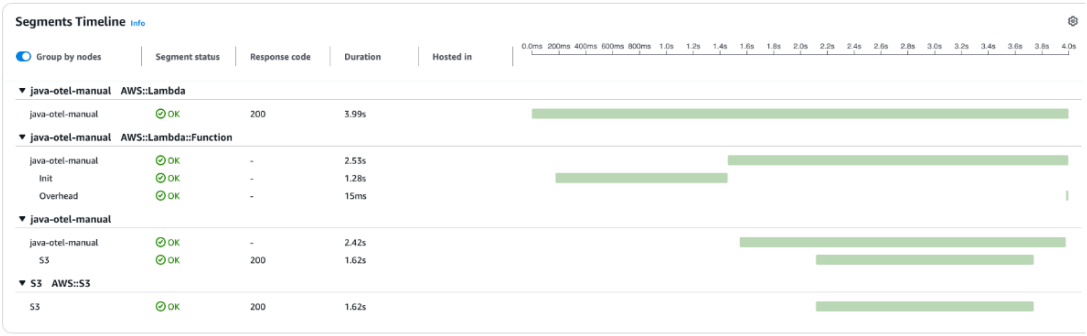
    @Override
    public String doHandleRequest(String input, Context context) {
        try {
            ListBucketsResponse response = makeListBucketsCall();
            context.getLogger().log("response: " + response.toString());
            return "Success";
        } catch (Exception e) {
            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);
```

```
        }
    }
}
```

Quando si richiama la funzione Lambda, verrà visualizzata la seguente traccia in Trace Map nella CloudWatch console.





Esegui la migrazione a OpenTelemetry Go

Usa i seguenti esempi di codice per strumentare manualmente le tue applicazioni Go con l' OpenTelemetry SDK durante la migrazione da X-Ray.

Strumentazione manuale con l'SDK

Tracing setup with X-Ray SDK

Quando si utilizzava X-Ray SDK for Go, era necessario configurare i plug-in di servizio o le regole di campionamento locali prima di utilizzare la strumentazione del codice.

```
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

Configura l' OpenTelemetry SDK creando un'istanza e registrandolo come provider di tracciamento globale TracerProvider . Ti consigliamo di configurare i seguenti componenti:

- OTLP Trace Exporter: necessario per esportare le tracce nell'agente o nel raccoglitore CloudWatch OpenTelemetry
- X-Ray Propagator: necessario per propagare il contesto di traccia ai AWS servizi integrati con X-Ray
- X-Ray Remote Sampler: necessario per le richieste di campionamento che utilizzano le regole di campionamento a raggi X
- Rilevatori di risorse: per rilevare i metadati dell'host che esegue l'applicazione

```
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
}
```

Tracciamento delle richieste in arrivo (strumentazione del gestore HTTP)

With X-Ray SDK

Per strumentare un gestore HTTP con X-Ray, è stato utilizzato il metodo del gestore X-Ray per generare segmenti utilizzando. 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

Per strumentare un gestore HTTP OpenTelemetry, utilizzate il metodo NewHandler per racchiudere il codice OpenTelemetry del gestore originale.

```
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 per strumentazione Go v2

With X-Ray SDK

Per gestire le AWS richieste in uscita da AWS SDK, i vostri clienti hanno utilizzato la seguente strumentazione:

```
// Create a segment
ctx, root := xray.BeginSegment(context.TODO(), "AWSSDKV2_Dynamodb")
defer root.Close(nil)
```

```
cfg, err := config.LoadDefaultConfig(ctx, config.WithRegion("us-west-2"))
if err != nil {
    log.Fatalf("unable to load SDK config, %v", err)
}
// Instrumenting AWS SDK v2
awsv2.AWSV2Instrumentor(&cfg.APIOptions)
// Using the Config value, create the DynamoDB client
svc := dynamodb.NewFromConfig(cfg)
// Build the request with its input parameters
_, err = svc.ListTables(ctx, &dynamodb.ListTablesInput{
    Limit: aws.Int32(5),
})
if err != nil {
    log.Fatalf("failed to list tables, %v", err)
}
```

With OpenTelemetry SDK

Il supporto di tracciamento per le chiamate AWS SDK downstream è fornito da SDK for AWS Go OpenTelemetry v2 Instrumentation. Ecco un esempio di tracciamento di una chiamata client 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
    }
```

Analisi delle chiamate HTTP in uscita

With X-Ray SDK

Per strumentare le chiamate HTTP in uscita con X-Ray, XRay.Client è stato utilizzato per creare una copia di un client HTTP fornito.

```
myClient := xray.Client(http-client)

resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

With OpenTelemetry SDK

Per strumentare i client HTTP, usa's otelhttp. OpenTelemetry OpenTelemetry NewTransport metodo per avvolgere l'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)
```

Supporto alla strumentazione per altre librerie

Puoi trovare l'elenco completo delle strumentazioni di libreria supportate per OpenTelemetry Go nella sezione Pacchetti Instrumentation.

In alternativa, puoi cercare OpenTelemetry nel registro per scoprire se OpenTelemetry supporta la strumentazione per la tua libreria sotto Registry.

Creazione manuale di dati di traccia

With X-Ray SDK

Con X-Ray SDK, erano necessari i BeginSubsegment metodi BeginSegment and per creare manualmente segmenti e sottosegmenti X-Ray.

```
// 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

Utilizza intervalli personalizzati per monitorare le prestazioni delle attività interne che non vengono acquisite dalle librerie di strumentazione. Nota che solo gli intervalli di tipo Server vengono convertiti in segmenti X-Ray, tutti gli altri intervalli vengono convertiti in sottosegmenti X-Ray.

Innanzitutto, è necessario creare un Tracer per generare intervalli, che è possibile ottenere tramite il metodo. `otel.Tracer` Ciò fornirà un'istanza Tracer tra quelle registrate globalmente nell'esempio di Tracing Setup. TracerProvider È possibile creare tutte le istanze Tracer necessarie, ma è comune avere una sola istanza Tracer per un'intera applicazione.

```
    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()
```

Aggiungere annotazioni e metadati alle tracce con SDK OpenTelemetry

Nell'esempio precedente, il `WithAttributes` metodo viene utilizzato per aggiungere attributi a ogni intervallo. Si noti che per impostazione predefinita, tutti gli attributi span vengono convertiti in metadati nei dati grezzi di X-Ray. Per garantire che un attributo venga convertito in un'annotazione e non in metadati, aggiungete la chiave dell'attributo all'elenco dell'attributo. `aws.xray.annotations` Per ulteriori informazioni, vedere Abilitare le annotazioni X-Ray personalizzate.

Strumentazione manuale Lambda

With X-Ray SDK

Con l'SDK X-Ray, dopo aver abilitato l'Active Tracing in Lambda, non erano necessarie configurazioni aggiuntive per utilizzare l'SDK X-Ray. Lambda ha creato un segmento che rappresenta l'invocazione del gestore Lambda e tu hai creato dei sottosegmenti utilizzando X-Ray SDK senza alcuna configurazione aggiuntiva.

With OpenTelemetry SDK

Il seguente codice di funzione Lambda (senza strumentazione) effettua una chiamata Amazon S3 e una richiesta HTTP in uscita 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)
}
```

Per strumentare manualmente il gestore Lambda e il client Amazon S3, procedi come segue:

1. In main (), crea un'istanza di a TracerProvider (tp) e registralo come fornitore globale di tracer. Si TracerProvider consiglia di configurarlo con:
   a. Simple Span Processor con un esportatore di span UDP a raggi X per inviare tracce all'endpoint UDP X-Ray di Lambda
   b. Risorsa con service.name impostato sul nome della funzione Lambda
2. Cambia l'utilizzo di to. `lambda.Start(lambdaHandler)` `lambda.Start(otellambda.InstrumentHandler(lambdaHandler, xrayconfig.WithRecommendedOptions(tp)...))`
3. Strumenta il client Amazon S3 con la strumentazione OpenTemetry AWS SDK aggiungendo OpenTelemetry middleware per `aws-sdk-go-v2` nella configurazione del client Amazon S3.
4. Strumenta il client http utilizzando il metodo's per avvolgere il. OpenTelemetry `otelhttp.NewTransport` `http.DefaultTransport`

Il codice seguente è un esempio di come apparirà la funzione Lambda dopo le modifiche. È possibile creare manualmente intervalli personalizzati aggiuntivi oltre agli intervalli forniti 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(otellambda.InstrumentHandler(lambdaHandler,
 xrayconfig.WithRecommendedOptions(tp)...))
```

```
}
```

Quando richiami Lambda, vedrai la seguente traccia `Trace Map` nella CloudWatch console:



Migrare a Node.js OpenTelemetry

Questa sezione spiega come migrare le applicazioni Node.js da X-Ray SDK a. OpenTelemetry Descrive gli approcci di strumentazione sia automatici che manuali e fornisce esempi specifici per casi d'uso comuni.

L'SDK X-Ray Node.js consente di strumentare manualmente le applicazioni Node.js per il tracciamento. Questa sezione fornisce esempi di codice per la migrazione da OpenTelemetry X-Ray alla strumentazione.

Sections

- Soluzioni di strumentazione automatica a codice zero
- Soluzioni di strumentazione manuale
- Tracciamento delle richieste in arrivo
- AWS Strumentazione SDK V3 JavaScript
- Analisi delle chiamate HTTP in uscita
- Supporto alla strumentazione per altre librerie
- Creazione manuale di dati di traccia
- Strumentazione Lambda

Soluzioni di strumentazione automatica a codice zero

Per tracciare le richieste con X-Ray SDK per Node.js, è necessario modificare il codice dell'applicazione. Con OpenTelemetry, puoi utilizzare soluzioni di strumentazione automatica a codice zero per tracciare le richieste.

Strumentazione automatica a codice zero con strumentazione automatica basata. OpenTelemetry

1. Utilizzo della strumentazione automatica AWS Distro for OpenTelemetry (ADOT) per Node.js: per la strumentazione automatica per l'applicazione Node.js, vedere Tracciamento e metriche con la distribuzione per la strumentazione automatica. AWS OpenTelemetry JavaScript

   (Facoltativo) È inoltre possibile abilitare CloudWatch Application Signals quando si attivano automaticamente le applicazioni AWS con la strumentazione JavaScript automatica ADOT per monitorare lo stato attuale delle applicazioni e tenere traccia delle prestazioni delle applicazioni a lungo termine rispetto agli obiettivi aziendali. Application Signals ti offre una visione unificata e incentrata sulle applicazioni di applicazioni, servizi e dipendenze e ti aiuta a monitorare e valutare lo stato delle applicazioni. Per ulteriori informazioni, consulta Application Signals.
2. Utilizzo della strumentazione automatica a OpenTelemetry JavaScript codice zero: per la strumentazione automatica con, vedere strumentazione a codice zero. OpenTelemetry JavaScript JavaScript

Soluzioni di strumentazione manuale

Tracing setup with X-Ray SDK

Quando si utilizzava X-Ray SDK per Node.js, era necessario configurare il `aws-xray-sdk` pacchetto X-Ray SDK con plug-in di servizio o regole di campionamento locali prima di utilizzare l'SDK per strumentare il codice.

```
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 Il campionamento remoto a raggi X non è attualmente disponibile per la configurazione per JS. OpenTelemetry Tuttavia, il supporto per il campionamento remoto a raggi X è attualmente disponibile tramite ADOT Auto-Instrumentation for Node.js.

Per l'esempio di codice riportato di seguito, sono necessarie le seguenti dipendenze:

```
npm install --save \
  @opentelemetry/api \
  @opentelemetry/sdk-node \
  @opentelemetry/exporter-trace-otlp-proto \
  @opentelemetry/propagator-aws-xray \
  @opentelemetry/resource-detector-aws
```

È necessario impostare e configurare l' OpenTelemetry SDK prima di eseguire il codice dell'applicazione. Questa operazione può essere eseguita utilizzando il flag —-require. Create un file chiamato instrumentation.js, che conterrà la configurazione e la configurazione della OpenTelemetry strumentazione.

Si consiglia di configurare i seguenti componenti:

- OTLPTraceEsportatore: necessario per esportare le tracce nell'Agent/ Collector CloudWatch OpenTelemetry
- AWSXRayPropagatore: necessario per propagare il Trace Context ai AWS servizi integrati con X-Ray
- Rilevatori di risorse (ad esempio, Amazon EC2 Resource Detector): per rilevare i metadati dell'host che esegue l'applicazione

```
/*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();
```

Quindi, puoi eseguire l'applicazione con la tua configurazione, ad esempio: OpenTelemetry

```
node --require ./instrumentation.js app.js
```

Puoi utilizzare la strumentazione della libreria OpenTelemetry SDK per creare automaticamente intervalli per librerie come l'SDK. AWS Abilitandoli si creeranno automaticamente degli intervalli per moduli come l'SDK per la v3. AWS JavaScript OpenTelemetry offre la possibilità di abilitare tutte le strumentazioni della libreria o specificare quali strumentazioni della libreria abilitare.

Per abilitare tutte le strumentazioni, installa il pacchetto: `@opentelemetry/auto-instrumentations-node`

```
npm install @opentelemetry/auto-instrumentations-node
```

Successivamente, aggiorna la configurazione per abilitare tutte le strumentazioni della libreria come mostrato di seguito.

```
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

È possibile utilizzare la strumentazione automatica ADOT per Node.js OpenTelemetry per configurare automaticamente le applicazioni Node.js. Utilizzando ADOT Auto-Instrumentation, non è necessario apportare modifiche manuali al codice per tracciare le richieste in arrivo o tracciare librerie come i client SDK o HTTP. AWS Per ulteriori informazioni, consulta Tracciamento e metriche con la distribuzione per Auto-Instrumentation. AWS OpenTelemetry JavaScript

La strumentazione automatica ADOT per Node.js supporta:

- Campionamento remoto a raggi X tramite variabile di ambiente — `export OTEL_TRACES_SAMPLER=xray`
- Propagazione contestuale delle tracce X-Ray (abilitata per impostazione predefinita)
- Rilevamento delle risorse (il rilevamento delle risorse per gli ambienti Amazon EC2, Amazon ECS e Amazon EKS è abilitato per impostazione predefinita)
- Strumentazioni di libreria automatiche per tutte le strumentazioni supportate, che possono essere OpenTelemetry utilizzate selettivamente tramite variabili di ambiente disabled/enabled `OTEL_NODE_ENABLED_INSTRUMENTATIONS` `OTEL_NODE_DISABLED_INSTRUMENTATIONS`
- Creazione manuale di Spans

Tracciamento delle richieste in arrivo

With X-Ray SDK

Express.js

Grazie all'SDK X-Ray per tracciare le richieste HTTP in entrata ricevute dalle applicazioni Express.js, i due middleware `AWSXRay.express.openSegment(<name>)` e i due middleware `AWSXRay.express.closeSegment()` dovevano avvolgere tutte le rotte definite per poterle tracciare.

```
app.use(xrayExpress.openSegment('defaultName'));

...

app.use(xrayExpress.closeSegment());
```

Restifica

Per tracciare le richieste HTTP in entrata ricevute dalle `Restify` applicazioni, è stato utilizzato il middleware dell'X-Ray SDK eseguendo enable dal modulo sul `aws-xray-sdk-restify` server 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

Il supporto di tracciamento per le richieste in entrata di `Express.js` è fornito dalla strumentazione OpenTelemetry HTTP e dalla strumentazione express. OpenTelemetry Installa le seguenti dipendenze con: `npm`

```
npm install --save @opentelemetry/instrumentation-http @opentelemetry/
instrumentation-express
```

Aggiorna la configurazione OpenTelemetry SDK per abilitare la strumentazione per il modulo express:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
const { ExpressInstrumentation } = require('@opentelemetry/instrumentation-
express');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    // Express instrumentation requires HTTP instrumentation
    new HttpInstrumentation(),
    new ExpressInstrumentation(),
  ],
});
```

Restifica

Per le applicazioni Restify, avrai bisogno della strumentazione OpenTelemetry Restify. Installa la seguente dipendenza:

```
npm install --save @opentelemetry/instrumentation-restify
```

Aggiorna la configurazione OpenTelemetry SDK per abilitare la strumentazione per il modulo restify:

```
const { RestifyInstrumentation } = require('@opentelemetry/instrumentation-
restify');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
```

```
    ...
    new RestifyInstrumentation(),
  ],
});
```

AWS Strumentazione SDK V3 JavaScript

With X-Ray SDK

Per strumentare AWS le richieste in uscita dall' AWS SDK, avete utilizzato client strumentati come nell'esempio seguente:

```
import { S3, PutObjectCommand } from '@aws-sdk/client-s3';

const s3 = AWSXRay.captureAWSv3Client(new S3({}));

await s3.send(new PutObjectCommand({
  Bucket: bucketName,
  Key: keyName,
  Body: 'Hello!',
}));
```

With OpenTelemetry SDK

Il supporto di tracciamento per le chiamate AWS SDK downstream a DynamoDB, Amazon S3 e altri è fornito dalla strumentazione SDK. OpenTelemetry AWS npmInstalla la seguente dipendenza con:

```
npm install --save @opentelemetry/instrumentation-aws-sdk
```

Aggiorna la configurazione OpenTelemetry SDK con la strumentazione AWS SDK.

```
import { AwsInstrumentation } from '@opentelemetry/instrumentation-aws-sdk';
...
```

```
const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new AwsInstrumentation()
  ],
});
```

Analisi delle chiamate HTTP in uscita

With X-Ray SDK

Per strumentare le richieste HTTP in uscita con X-Ray, era necessario strumentare i client. Ad esempio, vedi sotto.

Client HTTP individuali

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Tutti i client HTTP (globali)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

With OpenTelemetry SDK

Il supporto di tracciamento per i client HTTP Node.js è fornito da OpenTelemetry HTTP Instrumentation. Installa la seguente dipendenza con: `npm`

```
npm install --save @opentelemetry/instrumentation-http
```

Aggiorna la configurazione OpenTelemetry SDK come segue:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new HttpInstrumentation(),
  ],
});
```

Supporto alla strumentazione per altre librerie

Puoi trovare l'elenco completo delle strumentazioni di libreria supportate nella sezione Strumentazioni supportate OpenTelemetry JavaScript .

In alternativa, puoi cercare OpenTelemetry nel registro per scoprire se OpenTelemetry supporta la strumentazione per la tua libreria alla voce Registry.

Creazione manuale di dati di traccia

With X-Ray SDK

Utilizzando X-Ray, il codice del `aws-xray-sdk` pacchetto era necessario per creare manualmente i segmenti e i relativi sottosegmenti secondari per tracciare l'applicazione.

```
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

È possibile creare e utilizzare intervalli personalizzati per monitorare le prestazioni delle attività interne che non vengono acquisite dalle librerie di strumentazione. Nota che solo gli intervalli di tipo Server vengono convertiti in segmenti X-Ray, tutti gli altri intervalli vengono convertiti in sottosegmenti X-Ray. Per ulteriori informazioni, consulta Segmenti.

Avrai bisogno di un'istanza Tracer dopo aver configurato l'SDK in Tracing Setup per creare Spans OpenTelemetry . È possibile creare tutte le istanze Tracer necessarie, ma è normale avere una sola istanza Tracer per un'intera applicazione.

```
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();
  });
```

Aggiungere annotazioni e metadati alle tracce con SDK OpenTelemetry

Puoi anche aggiungere coppie chiave-valore personalizzate come attributi nei tuoi span. Nota che per impostazione predefinita, tutti questi attributi span verranno convertiti in metadati nei

dati grezzi di X-Ray. Per garantire che un attributo venga convertito in un'annotazione e non in metadati, aggiungete la chiave dell'attributo all'elenco dell'attributo. `aws.xray.annotations` Per ulteriori informazioni, vedere Abilitare le annotazioni X-Ray personalizzate.

```
 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();
 });
```

Strumentazione Lambda

With X-Ray SDK

Dopo aver abilitato Active Tracing per la funzione Lambda, era necessario l'SDK X-Ray senza alcuna configurazione aggiuntiva. Lambda crea un segmento che rappresenta l'invocazione del gestore Lambda e hai creato sottosegmenti o librerie di strumenti utilizzando l'SDK X-Ray senza alcuna configurazione aggiuntiva.

With OpenTelemetry SDK

Puoi strumentare automaticamente la tua Lambda con livelli Lambda AWS venduti. Esistono due soluzioni:

- (Consigliato) AWS Lambda Layer per OpenTelemetry

> **Note**
>
> Questo livello Lambda dispone di CloudWatch Application Signals abilitati per impostazione predefinita, il che consente il monitoraggio delle prestazioni e dello stato dell'applicazione Lambda raccogliendo sia metriche che tracce. Se vuoi solo tracciare, devi impostare la variabile di ambiente Lambda.

`OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false` Per ulteriori informazioni, consulta Abilita le tue applicazioni su Lambda.

- AWS layer Lambda gestito per ADOT JS. Per ulteriori informazioni, consulta AWS Distro for OpenTelemetry Lambda Support JavaScript For.

Creazione manuale di Span con strumentazione Lambda

Sebbene ADOT JavaScript Lambda Layer fornisca la strumentazione automatica per la funzione Lambda, potresti riscontrare la necessità di eseguire la strumentazione manuale nella tua Lambda, ad esempio, per fornire dati personalizzati o per il codice dello strumento all'interno della funzione Lambda stessa che non è coperto dalla strumentazione della libreria.

Per eseguire la strumentazione manuale insieme alla strumentazione automatica, è necessario aggiungerla come dipendenza. `@opentelemetry/api` Si consiglia che la versione di questa dipendenza sia la stessa della stessa dipendenza utilizzata da ADOT SDK. JavaScript Puoi utilizzare l' OpenTelemetry API per creare manualmente intervalli nella tua funzione Lambda.

Per aggiungere la `@opentelemetry/api` dipendenza usando NPM:

```
npm install @opentelemetry/api
```

Esegui la migrazione a.NET OpenTelemetry

Quando si utilizza X-Ray Tracing nelle applicazioni.NET, per la strumentazione viene utilizzato l'SDK X-Ray .NET con operazioni manuali.

Questa sezione fornisce esempi di codice nella Soluzioni di strumentazione manuale con SDK sezione per la migrazione dalla soluzione di strumentazione manuale X-Ray OpenTelemetry alle soluzioni di strumentazione manuale per.NET. In alternativa, è possibile migrare dalla strumentazione manuale X-Ray alle soluzioni di strumentazione automatica OpenTelemetry alle applicazioni.NET dello strumento senza dover modificare il codice sorgente dell'applicazione nella sezione. Soluzioni di strumentazione automatica a codice zero

Sections

- Soluzioni di strumentazione automatica a codice zero
- Soluzioni di strumentazione manuale con SDK
- Creazione manuale di dati di traccia
- Tracciamento delle richieste in arrivo (strumentazione di base ASP.NET e ASP.NET)
- AWS Strumentazione SDK
- Analisi delle chiamate HTTP in uscita
- Supporto alla strumentazione per altre librerie
- Strumentazione Lambda

Soluzioni di strumentazione automatica a codice zero

OpenTelemetry fornisce soluzioni di strumentazione automatica a codice zero. Queste soluzioni tracciano le richieste senza richiedere modifiche al codice dell'applicazione.

OpenTelemetryopzioni di strumentazione automatica basate su

1. Utilizzo della strumentazione automatica AWS Distro for OpenTelemetry (ADOT) per .NET: per strumentare automaticamente le applicazioni.NET, vedere Tracciamento e metriche con la distribuzione per strumentazione automatica.NET. AWS OpenTelemetry

   (Facoltativo) Abilita i segnali CloudWatch applicativi durante la strumentazione automatica delle applicazioni AWS con la strumentazione automatica ADOT .NET per:

   - Monitora lo stato attuale delle applicazioni
   - Monitora le prestazioni delle applicazioni a lungo termine rispetto agli obiettivi aziendali
   - Ottieni una visione unificata e incentrata sulle applicazioni delle tue applicazioni, dei tuoi servizi e delle tue dipendenze
   - Monitora e valuta lo stato delle applicazioni

   Per ulteriori informazioni, consulta Application Signals.
2. Utilizzo della strumentazione OpenTelemetry automatica.Net a codice zero: per strumentare automaticamente con OpenTelemetry .Net, vedi Tracciamento e metriche con la distribuzione per la strumentazione automatica.NET. AWS OpenTelemetry

Soluzioni di strumentazione manuale con SDK

Tracing configuration with X-Ray SDK

Per le applicazioni Web.NET, l'SDK X-Ray è configurato nella sezione AppSettings del file. `Web.config`

Esempio Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
  </appSettings>
</configuration>
```

Per .NET Core, viene utilizzato un file denominato `appsettings.json` con una chiave di primo livello denominata`XRay`, quindi viene creato un oggetto di configurazione per inizializzare il registratore X-Ray.

Esempio per.NET `appsettings.json`

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin"
  }
}
```

Esempio di configurazione di.NET Core Program.cs — Recorder

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Tracing configuration with OpenTelemetry SDK

Aggiungi queste dipendenze:

```
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
```

Per la tua applicazione.NET, configura l' OpenTelemetry SDK configurando Global. TracerProvider La configurazione di esempio seguente abilita anche la strumentazione per. `ASP.NET Core` Per lo strumento`ASP.NET`, vedere. Tracciamento delle richieste in arrivo (strumentazione di base ASP.NET e ASP.NET) Per utilizzarlo OpenTelemetry con altri framework, consulta Registry per ulteriori librerie per i framework supportati.

Si consiglia di configurare i seguenti componenti:

- `An OTLP Exporter`— Necessario per esportare le tracce nell' CloudWatch OpenTelemetry Agent/ Collector
- Un propagatore AWS a raggi X: necessario per propagare il Trace Context ai AWS servizi integrati con X-Ray
- Un campionatore remoto AWS a raggi X: necessario se è necessario campionare le richieste utilizzando le regole di campionamento a raggi X
- `Resource Detectors`(ad esempio, Amazon EC2 Resource Detector): per rilevare i metadati dell'host che esegue l'applicazione

```
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
```

Da utilizzare OpenTelemetry per un'app per console, aggiungi la seguente OpenTelemetry configurazione all'avvio del programma.

```
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
```

Creazione manuale di dati di traccia

With X-Ray SDK

Con X-Ray SDK, erano necessari i `BeginSubsegment` metodi `BeginSegment` and per creare manualmente segmenti e sottosegmenti 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

In .NET, è possibile utilizzare l'API Activity per creare intervalli personalizzati per monitorare le prestazioni delle attività interne che non vengono acquisite dalle librerie di strumentazione. Nota che solo gli intervalli di tipo Server vengono convertiti in segmenti X-Ray, tutti gli altri intervalli vengono convertiti in sotto-segmenti X-Ray.

È possibile creare tutte `ActivitySource` le istanze necessarie, ma si consiglia di averne solo una per un'intera applicazione/servizio.

```
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
```

```
    }
}
```

Aggiungere annotazioni e metadati alle tracce con SDK OpenTelemetry

Puoi anche aggiungere coppie chiave-valore personalizzate come attributi agli intervalli utilizzando il metodo su un'attività. `SetTag` Nota che per impostazione predefinita, tutti gli attributi span verranno convertiti in metadati nei dati grezzi di X-Ray. Per garantire che un attributo venga convertito in un'annotazione e non in metadati, puoi aggiungere la chiave di quell'attributo all'elenco degli attributi. `aws.xray.annotations`

```
using (var activity = activitySource.StartActivity("ActivityName",
 ActivityKind.Server)) // this will be translated to a X-Ray Segment
{
    activity.SetTag("metadataKey", "metadataValue");
    activity.SetTag("annotationKey", "annotationValue");
    string[] annotationKeys = {"annotationKey"};
    activity.SetTag("aws.xray.annotations", annotationKeys);

    // Do something here

    using (var internalActivity = activitySource.StartActivity("ActivityName",
 ActivityKind.Internal)) // this will be translated to an X-Ray Subsegment
    {
        // Do something here
    }
}
```

Con OpenTelemetry strumentazione automatica

Se si utilizza una soluzione di strumentazione OpenTelemetry automatica per.NET e se è necessario eseguire la strumentazione manuale nell'applicazione, ad esempio, per codificare lo strumento all'interno dell'applicazione stessa per sezioni che non sono coperte da alcuna libreria di strumentazione automatica.

Poiché può esserci solo una strumentazione globale`TracerProvider`, la strumentazione manuale non dovrebbe istanziare la propria `TracerProvider` se utilizzata insieme alla strumentazione automatica. Quando `TracerProvider` viene utilizzato, il tracciamento manuale

personalizzato funziona allo stesso modo quando si utilizza la strumentazione automatica o la strumentazione manuale tramite l'SDK. OpenTelemetry

Tracciamento delle richieste in arrivo (strumentazione di base ASP.NET e ASP.NET)

With X-Ray SDK

Per informazioni sulle richieste di strumenti servite dall'applicazione ASP.NET, consulta https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-dotnet-messagehandler.html per informazioni su come `RegisterXRay` richiamare il `Init` metodo del file. `global.asax`

```
AWSXRayASPNET.RegisterXRay(this, "MyApp");
```

Per le richieste di strumenti servite dall'applicazione principale ASP.NET, il `UseXRay` metodo viene chiamato prima di qualsiasi altro middleware nel `Configure` metodo della classe Startup.

```
app.UseXRay("MyApp");
```

With OpenTelemetry SDK

OpenTelemetry fornisce inoltre librerie di strumentazione per raccogliere tracce per le richieste Web in entrata per ASP.NET e ASP.NET core. La sezione seguente elenca i passaggi necessari per aggiungere e abilitare questi strumenti di libreria per la OpenTelemetry configurazione, incluso come aggiungere la strumentazione di base ASP.NET o ASP.NET durante la creazione del Tracer Provider.

Per informazioni su come abilitare .Instrumentation. OpenTelemetry AspNet, vedi Passaggi per abilitare .Instrumentation. OpenTelemetry AspNete per informazioni su come abilitare .Instrumentation. OpenTelemetry AspNetCore, vedi Passaggi per abilitare .Instrumentation. OpenTelemetry AspNetCore.

AWS Strumentazione SDK

With X-Ray SDK

Installa tutti i client AWS SDK chiamando. `RegisterXRayForAllServices()`

```
using Amazon.XRay.Recorder.Handlers.AwsSdk;
AWSSDKHandler.RegisterXRayForAllServices(); //place this before any instantiation of
 AmazonServiceClient
AmazonDynamoDBClient client = new AmazonDynamoDBClient(RegionEndpoint.USWest2); //
 AmazonDynamoDBClient is automatically registered with X-Ray
```

Utilizzate uno dei seguenti metodi per la strumentazione specifica del client AWS di servizio.

```
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

Per il seguente esempio di codice, è necessaria la seguente dipendenza:

```
dotnet add package OpenTelemetry.Instrumentation.AWS
```

Per strumentare l' AWS SDK, aggiorna la configurazione OpenTelemetry SDK in cui è configurato il Global TracerProvider .

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
        .AddAWSInstrumentation()
        ...
```

Analisi delle chiamate HTTP in uscita

With X-Ray SDK

X-Ray .NET SDK traccia le chiamate HTTP in uscita tramite i metodi di estensione `GetResponseTraced()` o `GetAsyncResponseTraced()` durante l'utilizzo o utilizzando `System.Net.HttpWebRequest` il gestore durante l'`HttpClientXRayTracingHandler`utilizzo. `System.Net.Http.HttpClient`

With OpenTelemetry SDK

Per il seguente esempio di codice, è necessaria la seguente dipendenza:

```
dotnet add package OpenTelemetry.Instrumentation.Http
```

Per strumentare `System.Net.Http.HttpClient` e`System.Net.HttpWebRequest`, aggiorna la configurazione dell' OpenTelemetry SDK in cui TracerProvider è configurato il Global.

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
        .AddHttpClientInstrumentation()
        ...
```

Supporto alla strumentazione per altre librerie

Puoi cercare e filtrare OpenTelemetry nel Registro le librerie di strumentazione.NET per scoprire se OpenTelemetry supporta la strumentazione per la tua libreria. Consulta il Registro per iniziare la ricerca.

Strumentazione Lambda

With X-Ray SDK

Per utilizzare l'SDK X-Ray con Lambda, era necessaria la seguente procedura:

1. Abilita Active Tracing sulla tua funzione Lambda
2. Il servizio Lambda crea un segmento che rappresenta la chiamata del gestore
3. Crea sottosegmenti o librerie di strumenti utilizzando l'SDK X-Ray

With OpenTelemetry-based solutions

Puoi strumentare automaticamente la tua Lambda con livelli Lambda AWS venduti. Esistono due soluzioni:

- (Consigliato) AWS Lambda Layer per OpenTelemetry
- Per prestazioni migliori, potresti prendere in considerazione l'idea di utilizzare per `OpenTelemetry Manual Instrumentation` generare OpenTelemetry tracce per la tua funzione Lambda.

OpenTelemetry strumentazione manuale per Lambda AWS

Di seguito è riportato l'esempio del codice della funzione Lambda (senza strumentazione).

```
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);
        }
    }
}
```

Per strumentare manualmente il tuo gestore Lambda e il client Amazon S3, procedi come segue.

1. Istanzia a TracerProvider : si TracerProvider consiglia di configurarlo con un `XrayUdpSpanExporter` campionatore ParentBased Always On e un `Resource` valore impostato sul nome della funzione `service.name` Lambda.
2. Strumenta il client Amazon S3 con la strumentazione OpenTemetry AWS SDK chiamando per aggiungere la strumentazione client SDK `AddAWSInstrumentation()` a AWS `TracerProvider`
3. Crea una funzione wrapper con la stessa firma della funzione Lambda originale. Chiama `AWSLambdaWrapper.Trace()` l'API e passa`TracerProvider`, la funzione Lambda originale e i suoi input come parametri. Imposta la funzione wrapper come input del gestore Lambda.

Per il seguente esempio di codice, sono necessarie le seguenti dipendenze:

```
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
```

Il codice seguente illustra la funzione Lambda dopo le modifiche richieste. È possibile creare intervalli personalizzati aggiuntivi per completare gli intervalli forniti 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);
        }
    }
}
```

Quando richiami questa Lambda, vedrai la seguente traccia nella Trace Map della console: CloudWatch



Migrare a Python OpenTelemetry

Questa guida ti aiuta a migrare le applicazioni Python da X-Ray SDK alla strumentazione. OpenTelemetry Copre approcci di strumentazione sia automatici che manuali, con esempi di codice per scenari comuni.

Sections

- Soluzioni di strumentazione automatica a codice zero
- Strumenta manualmente le tue applicazioni
- Tracciamento dell'inizializzazione della configurazione
- Tracciamento delle richieste in arrivo
- AWS Strumentazione SDK
- Strumentazione delle chiamate HTTP in uscita tramite richieste
- Supporto strumentale per altre librerie

- Creazione manuale di dati di traccia
- Strumentazione Lambda

Soluzioni di strumentazione automatica a codice zero

Con X-Ray SDK, era necessario modificare il codice dell'applicazione per tracciare le richieste. OpenTelemetry offre soluzioni di strumentazione automatica a codice zero per tracciare le richieste. Con OpenTelemetry, hai la possibilità di utilizzare soluzioni di strumentazione automatica a codice zero per tracciare le richieste.

Codice zero con strumentazione automatica basata OpenTelemetry

1. Utilizzo della strumentazione automatica AWS Distro for OpenTelemetry (ADOT) per Python — Per la strumentazione automatica per le applicazioni Python, vedi Tracing and Metrics with the Distro for Python Auto-Instrumentation. AWS OpenTelemetry

   (Facoltativo) È inoltre possibile abilitare CloudWatch Application Signals durante la strumentazione automatica delle applicazioni AWS con la strumentazione automatica ADOT Python per monitorare lo stato attuale delle applicazioni e tenere traccia delle prestazioni delle applicazioni a lungo termine rispetto agli obiettivi aziendali. Application Signals ti offre una visione unificata e incentrata sulle applicazioni di applicazioni, servizi e dipendenze e ti aiuta a monitorare e valutare lo stato delle applicazioni.
2. Utilizzo della strumentazione automatica OpenTelemetry Python a codice zero — Per la strumentazione automatica con Python, vedi Strumentazione Python a codice zero. OpenTelemetry

Strumenta manualmente le tue applicazioni

È possibile strumentare manualmente le applicazioni utilizzando il 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
```

Tracciamento dell'inizializzazione della configurazione

With X-Ray SDK

In X-Ray, il globale `xray_recorder` viene inizializzato e utilizzato per generare segmenti e sottosegmenti.

With OpenTelemetry SDK

> **Note**
>
> X-Ray Remote Sampling non è attualmente disponibile per la configurazione per Python. OpenTelemetry Tuttavia, il supporto per il campionamento remoto a raggi X è attualmente disponibile tramite ADOT Auto-Instrumentation for Python.

Nel, è necessario inizializzare un OpenTelemetry file globale. `TracerProvider` In questo modo`TracerProvider`, è possibile acquisire un Tracer che è possibile utilizzare per generare intervalli in qualsiasi punto dell'applicazione. Si consiglia di configurare i seguenti componenti:

- `OTLPSpanExporter`— Necessario per esportare le tracce nell' CloudWatch OpenTelemetry Agent/ Collector
- Un propagatore AWS a raggi X: necessario per propagare il Trace Context ai AWS servizi integrati con X-Ray

```
from opentelemetry import (
    trace,
    propagate
)
from opentelemetry.sdk.trace import TracerProvider
```

```
from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.exporter.otlp.proto.http.trace_exporter import OTLPSpanExporter
from opentelemetry.sdk.trace.export import BatchSpanProcessor
from opentelemetry.propagators.aws import AwsXRayPropagator

# Sends generated traces in the OTLP format to an OTel Collector running on port
 4318
otlp_exporter = OTLPSpanExporter(endpoint="http://localhost:4318/v1/traces")
# Processes traces in batches as opposed to immediately one after the other
span_processor = BatchSpanProcessor(otlp_exporter)
# More configurations can be done here. We will visit them later.

# Sets the global default tracer provider
provider = TracerProvider(active_span_processor=span_processor)
trace.set_tracer_provider(provider)

# Configures the global propagator to use the X-Ray Propagator
propagate.set_global_textmap(AwsXRayPropagator())

# Creates a tracer from the global tracer provider
tracer = trace.get_tracer("my.tracer.name")
# Use this tracer to create Spans
```

Con strumentazione automatica ADOT per Python

Puoi usare la strumentazione automatica ADOT per Python per configurare OpenTelemetry automaticamente le tue applicazioni Python. Utilizzando la strumentazione automatica ADOT, non è necessario apportare modifiche manuali al codice per tracciare le richieste in arrivo o tracciare librerie come i AWS client SDK o HTTP. Per ulteriori informazioni, consulta Tracciamento e metriche con la distribuzione AWS per Python OpenTelemetry Auto-Instrumentation.

La strumentazione automatica ADOT per Python supporta:

- Campionamento remoto a raggi X tramite la variabile di ambiente `export OTEL_TRACES_SAMPLER=xray`
- Propagazione contestuale delle tracce X-Ray (abilitata per impostazione predefinita)
- Rilevamento delle risorse (il rilevamento delle risorse per gli ambienti Amazon EC2, Amazon ECS e Amazon EKS è abilitato per impostazione predefinita)

- Le strumentazioni di libreria automatiche per tutte le strumentazioni supportate sono abilitate per OpenTelemetry impostazione predefinita. È possibile disabilitarla in modo selettivo tramite la variabile di ambiente. `OTEL_PYTHON_DISABLED_INSTRUMENTATIONS` (tutti sono abilitati di default)
- Creazione manuale di Spans

Dai plug-in dei servizi X-Ray ai provider di risorse OpenTelemetry AWS

L'X-Ray SDK fornisce plug-in che è possibile aggiungere per acquisire informazioni specifiche sulla piattaforma dal servizio ospitato come Amazon EC2, Amazon ECS ed Elastic Beanstalk. `xray_recorder` È simile ai Resource Providers in quanto acquisisce le informazioni come attributi di risorse. OpenTelemetry Sono disponibili più Resource Provider per diverse AWS piattaforme.

- Inizia installando il pacchetto di AWS estensione, `pip install opentelemetry-sdk-extension-aws`
- Configura il rilevatore di risorse desiderato. L'esempio seguente mostra come configurare il provider di risorse Amazon EC2 in SDK. OpenTelemetry

```
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)
```

Tracciamento delle richieste in arrivo

With X-Ray SDK

L'X-Ray Python SDK supporta framework applicativi come Django, Flask e Bottle per tracciare le richieste in arrivo per le applicazioni Python in esecuzione su di essi. Questo viene fatto aggiungendo all'applicazione per ogni framework. `XRayMiddleware`

With OpenTelemetry SDK

OpenTelemetry fornisce strumentazioni per Django e Flask attraverso le librerie di strumentazione specifiche. Non è disponibile alcuna strumentazione per Bottle OpenTelemetry, le applicazioni possono ancora essere tracciate utilizzando la strumentazione WSGI. OpenTelemetry

Per il seguente esempio di codice, è necessaria la seguente dipendenza:

```
pip install opentelemetry-instrumentation-flask
```

È necessario inizializzare l' OpenTelemetry SDK e registrare quello globale TracerProvider prima di aggiungere strumentazioni per il framework dell'applicazione. Senza di essa, le operazioni di tracciamento lo saranno. `no-ops` Dopo aver configurato il globale`TracerProvider`, è possibile utilizzare lo strumentor per il framework dell'applicazione. L'esempio seguente mostra un'applicazione 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 Strumentazione SDK

With X-Ray SDK

L'X-Ray Python SDK traccia la richiesta del client SDK applicando una AWS patch alla libreria. `botocore` Per ulteriori informazioni, consulta Tracciare le chiamate AWS SDK con l'SDK X-Ray per Python. Nell'applicazione, il `patch_all()` metodo viene utilizzato per strumentare tutte le librerie o applicare le patch in modo selettivo utilizzando le librerie or. `botocore` `boto3` `patch(['botocore'])` Qualsiasi metodo scelto monitora tutti i client Boto3 dell'applicazione e genera un sottosegmento per ogni chiamata effettuata utilizzando questi client.

With OpenTelemetry SDK

Per il seguente esempio di codice, avrai bisogno della seguente dipendenza:

```
pip install opentelemetry-instrumentation-botocore
```

Utilizzate la strumentazione OpenTelemetry Botocore a livello di codice per strumentare tutti i client Boto3 della vostra applicazione. L'esempio seguente illustra `botocore` la strumentazione.

```
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}")
```

Strumentazione delle chiamate HTTP in uscita tramite richieste

With X-Ray SDK

L'X-Ray Python SDK traccia le chiamate HTTP in uscita tramite le richieste applicando patch alla libreria delle richieste. Per ulteriori informazioni, consulta Tracciare le chiamate ai servizi Web HTTP downstream utilizzando X-Ray SDK per Python. Nella tua applicazione, puoi utilizzare il `patch_all()` metodo per strumentare tutte le librerie o patchare selettivamente le librerie di richieste utilizzando. `patch(['requests'])` Qualsiasi opzione strumenta la `requests` libreria, generando un sottosegmento per ogni chiamata effettuata. `requests`

With OpenTelemetry SDK

Per il seguente esempio di codice, è necessaria la seguente dipendenza:

```
pip install opentelemetry-instrumentation-requests
```

OpenTelemetry Utilizzate la strumentazione delle richieste a livello di codice per strumentare la libreria delle richieste per generare tracce per le richieste HTTP da essa effettuate nell'applicazione. Per ulteriori informazioni, consulta request Instrumentation. OpenTelemetry L'esempio seguente illustra la strumentazione della `requests` libreria.

```
from opentelemetry.instrumentation.requests import RequestsInstrumentor

# Instrument Requests
RequestsInstrumentor().instrument()

...

    example_session = requests.Session()
    example_session.get(url="https://example.com")
```

In alternativa, puoi anche strumentare la `urllib3` libreria sottostante per tracciare le richieste 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")
```

Supporto strumentale per altre librerie

Puoi trovare l'elenco completo delle strumentazioni di libreria supportate per OpenTelemetry Python in Librerie, framework, server di applicazioni e. JVMs

In alternativa, puoi cercare OpenTelemetry nel Registro per scoprire se supporta la strumentazione. OpenTelemetry Consulta il Registro per iniziare la ricerca.

Creazione manuale di dati di traccia

Puoi creare segmenti e sottosegmenti usando la nella `xray_recorder` tua applicazione Python. Per ulteriori informazioni, consulta Strumentazione manuale del codice Python. Puoi anche aggiungere manualmente annotazioni e metadati ai dati di traccia.

Creazione di intervalli con SDK OpenTelemetry

Usa l'`start_as_current_span`API per avviare un intervallo e impostala per la creazione di intervalli. Per esempi sulla creazione di intervalli, consulta Creazione di intervalli. Una volta avviato un intervallo e rientrato nell'ambito corrente, è possibile aggiungervi ulteriori informazioni aggiungendo attributi, eventi, eccezioni, collegamenti e così via. Come abbiamo segmenti e sottosegmenti in X-Ray, ci sono diversi tipi di intervalli. OpenTelemetry Solo le campate `SERVER` tipo vengono convertite in segmenti a raggi X, mentre le altre vengono convertite in sottosegmenti a raggi X.

```
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
```

Aggiungere annotazioni e metadati alle tracce con SDK OpenTelemetry

L'X-Ray Python SDK fornisce informazioni separate APIs e consente di aggiungere annotazioni `put_annotation` e `put_metadata` metadati a una traccia. In OpenTelemetry SDK, le annotazioni e i metadati sono semplicemente attributi su un intervallo, aggiunti tramite l'API. `set_attribute`

Gli attributi Span che desideri siano annotazioni su una traccia vengono aggiunti sotto la chiave riservata il `aws.xray.annotations` cui valore è un elenco di coppie di annotazioni chiave-valore. Tutti gli altri attributi span diventano metadati sul segmento o sottosegmento convertito.

Inoltre, se si utilizza il collettore ADOT, è possibile configurare quali attributi span devono essere convertiti in annotazioni X-Ray specificando nella configurazione del collettore. `indexed_attributes`

L'esempio seguente mostra come aggiungere annotazioni e metadati a una traccia utilizzando 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")
```

Strumentazione Lambda

Per monitorare le funzioni lambda su X-Ray, abiliti X-Ray e hai aggiunto le autorizzazioni appropriate al ruolo di invocazione della funzione. Inoltre, se stai tracciando le richieste downstream della tua funzione, dovresti strumentalizzare il codice con X-Ray Python SDK.

OpenTelemetry Per X-Ray, si consiglia di utilizzare il AWS Lambda Layer per i segnali di applicazione OpenTelemetry disattivati. Questo strumenterà automaticamente la tua funzione e genererà intervalli per l'invocazione della funzione e qualsiasi richiesta derivante dalla tua funzione. Oltre al tracciamento, se sei interessato a utilizzare Application Signals per monitorare lo stato della tua funzione, consulta Abilitare le tue applicazioni su Lambda.

- Trova l'ARN del layer Lambda richiesto per la tua funzione da AWS Lambda Layer for e aggiungilo. OpenTelemetry ARNs
- Imposta le seguenti variabili di ambiente per la tua funzione.
  - `AWS_LAMBDA_EXEC_WRAPPER=/opt/otel-instrument`— Questo carica la strumentazione automatica per la funzione
  - `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`— Ciò disabiliterà il monitoraggio dei segnali applicativi

Creazione manuale di campate con la strumentazione Lambda

Inoltre, puoi generare intervalli personalizzati all'interno della tua funzione per tenere traccia del lavoro. È possibile farlo utilizzando solo il `opentelemetry-api` pacchetto in combinazione con il AWS Lambda Layer OpenTelemetry per la strumentazione automatica.

1. Includili `opentelemetry-api` come dipendenza nella tua funzione
2. Il seguente frammento di codice è un esempio per generare intervalli personalizzati

```
from opentelemetry import trace

# Get the tracer (auto-configured by the AWS Lambda Layer for OpenTelemetry)
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
    }
```

Migra a Ruby OpenTelemetry

Per migrare le applicazioni Ruby da X-Ray SDK alla OpenTelemetry strumentazione, utilizzate i seguenti esempi di codice e le linee guida per la strumentazione manuale.

Sections

- Strumenta manualmente le tue soluzioni con l'SDK
- Tracciamento delle richieste in arrivo (strumentazione Rails)
- AWS Strumentazione SDK
- Analisi delle chiamate HTTP in uscita
- Supporto alla strumentazione per altre librerie
- Creazione manuale dei dati di traccia
- Strumentazione manuale Lambda

Strumenta manualmente le tue soluzioni con l'SDK

Tracing setup with X-Ray SDK

X-Ray SDK for Ruby richiedeva la configurazione del codice con i plug-in di servizio.

```
require 'aws-xray-sdk'

XRay.recorder.configure(plugins: [:ec2, :elastic_beanstalk])
```

Tracing setup with OpenTelemetry SDK

> **Note**
>
> Il campionamento remoto a raggi X non è attualmente disponibile per la configurazione per Ruby. OpenTelemetry

Per un'applicazione Ruby on Rails, inserite il codice di configurazione in un inizializzatore Rails. Per ulteriori informazioni, consulta Nozioni di base su . Per tutti i programmi Ruby con strumentazione manuale, è necessario utilizzare il metodo per configurare l'`OpenTelemetry::SDK.configure`SDK Ruby. OpenTelemetry

Innanzitutto, installa i seguenti pacchetti:

```
bundle add opentelemetry-sdk opentelemetry-exporter-otlp opentelemetry-propagator-
xray
```

Quindi, configura l' OpenTelemetry SDK tramite il codice di configurazione che viene eseguito quando il programma viene inizializzato. Si consiglia di configurare i seguenti componenti:

- `OTLP Exporter`— Necessario per esportare le tracce nell' CloudWatch agente e OpenTelemetry nel raccoglitore
- `An AWS X-Ray Propagator`— Necessario per propagare il contesto di traccia ai AWS servizi integrati con X-Ray

```
require 'opentelemetry-sdk'
```

```
require 'opentelemetry-exporter-otlp'

# Import the gem containing the AWS X-Ray for OTel Ruby ID Generator and propagator
require 'opentelemetry-propagator-xray'

OpenTelemetry::SDK.configure do |c|
  c.service_name = 'my-service-name'

  c.add_span_processor(
    # Use the BatchSpanProcessor to send traces in groups instead of one at a time
    OpenTelemetry::SDK::Trace::Export::BatchSpanProcessor.new(
      # Use the default OLTP Exporter to send traces to the ADOT Collector
      OpenTelemetry::Exporter::OTLP::Exporter.new(
        # The OpenTelemetry Collector is running as a sidecar and listening on port
 4318
        endpoint:"http://127.0.0.1:4318/v1/traces"
      )
    )
  )

  # The X-Ray Propagator injects the X-Ray Tracing Header into downstream calls
  c.propagators = [OpenTelemetry::Propagator::XRay::TextMapPropagator.new]
end
```

OpenTelemetry SDKs hanno anche il concetto di strumentazione di libreria. Abilitandoli si creeranno automaticamente degli intervalli per librerie come l'SDK. AWS OpenTelemetry offre la possibilità di abilitare tutte le strumentazioni della libreria o specificare quali strumentazioni della libreria abilitare.

Per abilitare tutte le strumentazioni, installa prima il pacchetto: `opentelemetry-instrumentation-all`

```
bundle add opentelemetry-instrumentation-all
```

Successivamente, aggiorna la configurazione per abilitare tutte le strumentazioni della libreria come mostrato di seguito:

```
require 'opentelemetry/instrumentation/all'
...
```

```
OpenTelemetry::SDK.configure do |c|
   ...

  c.use_all() # Enable all instrumentations
end
```

OpenTelemetry SDKs hanno anche il concetto di strumentazione di libreria. Abilitandoli si creeranno automaticamente degli intervalli per librerie come l'SDK. AWS OpenTelemetry offre la possibilità di abilitare tutte le strumentazioni della libreria o specificare quali strumentazioni della libreria abilitare.

Per abilitare tutte le strumentazioni, installa prima il pacchetto: `opentelemetry-instrumentation-all`

```
bundle add opentelemetry-instrumentation-all
```

Successivamente, aggiorna la configurazione per abilitare tutte le strumentazioni della libreria come mostrato di seguito:

```
require 'opentelemetry/instrumentation/all'
...

OpenTelemetry::SDK.configure do |c|
   ...

  c.use_all() # Enable all instrumentations
end
```

Tracciamento delle richieste in arrivo (strumentazione Rails)

With X-Ray SDK

Con X-Ray SDK, il tracciamento X-Ray viene configurato per il framework Rails al momento dell'inizializzazione.

config/initializers/awsEsempio: _xray.rb

```
Rails.application.config.xray = {
  name: 'my app',
  patch: %I[net_http aws_sdk],
  active_record: true
}
```

With OpenTelemetry SDK

Innanzitutto, installa i seguenti pacchetti:

```
bundle add opentelemetry-instrumentation-rack opentelemetry-instrumentation-
rails opentelemetry-instrumentation-action_pack opentelemetry-instrumentation-
active_record opentelemetry-instrumentation-action_view
```

Successivamente, aggiorna la configurazione per abilitare la strumentazione per la tua applicazione Rails come mostrato di seguito:

```
# 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 Strumentazione SDK

With X-Ray SDK

Per strumentare AWS le richieste in uscita dall' AWS SDK, ai client AWS SDK viene applicata una patch con X-Ray come nell'esempio seguente:

```
require 'aws-xray-sdk'
require 'aws-sdk-s3'

# Patch AWS SDK clients
XRay.recorder.configure(plugins: [:aws_sdk])

# Use the instrumented client
s3 = Aws::S3::Client.new
s3.list_buckets
```

With OpenTelemetry SDK

AWS SDK for Ruby V3 fornisce supporto per la registrazione e l'emissione di tracce. OpenTelemetry Per informazioni su come configurare un client di servizio, consulta Configurazione delle funzionalità di osservabilità nell' AWS SDK OpenTelemetry for Ruby.

Analisi delle chiamate HTTP in uscita

Quando si effettuano chiamate HTTP a servizi esterni, potrebbe essere necessario strumentare manualmente le chiamate se la strumentazione automatica non è disponibile o non fornisce dettagli sufficienti.

With X-Ray SDK

Per strumentare le chiamate downstream, è stato utilizzato l'X-Ray SDK for Ruby per `net/http` applicare patch alla libreria utilizzata dall'applicazione:

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http]
}

XRay.recorder.configure(config)
```

With OpenTelemetry SDK

Per abilitare l'utilizzo OpenTelemetry della `net/http` strumentazione, installate innanzitutto il pacchetto: `opentelemetry-instrumentation-net_http`

```
bundle add opentelemetry-instrumentation-net_http
```

Quindi, aggiorna la configurazione per abilitare la `net/http` strumentazione come mostrato di seguito:

```
OpenTelemetry::SDK.configure do |c|
   ...

  c.use 'OpenTelemetry::Instrumentation::Net::HTTP'
  ...

end
```

Supporto alla strumentazione per altre librerie

Puoi trovare l'elenco completo delle strumentazioni di libreria supportate per Ruby sotto. OpenTelemetry opentelemetry-ruby-contrib

In alternativa, puoi cercare OpenTelemetry nel Registro per scoprire se OpenTelemetry supporta la strumentazione. Per ulteriori informazioni, consulta Registry.

Creazione manuale dei dati di traccia

With X-Ray SDK

Utilizzando X-Ray, il `aws-xray-sdk` pacchetto richiedeva la creazione manuale dei segmenti e dei relativi sottosegmenti secondari per tracciare l'applicazione. Potresti aver anche aggiunto annotazioni e metadati X-Ray ai tuoi segmenti o sottosegmenti:

```
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

È possibile utilizzare intervalli personalizzati per monitorare le prestazioni delle attività interne che non vengono acquisite dalle librerie di strumentazione. Nota che solo gli span di tipo server vengono convertiti in segmenti X-Ray, tutti gli altri intervalli vengono convertiti in sottosegmenti X-Ray. Per impostazione `INTERNAL` predefinita, gli intervalli sono.

Innanzitutto, crea un Tracer per generare intervalli, che puoi ottenere tramite il metodo. `OpenTelemetry.tracer_provider.tracer('<YOUR_TRACER_NAME>')` Ciò fornirà un'istanza Tracer registrata a livello globale nella configurazione dell'applicazione. OpenTelemetry È comune avere un singolo Tracer per un'intera applicazione. Crea un OpenTelemetry tracer e usalo per creare intervalli:

```
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
```

Aggiungere annotazioni e metadati alle tracce con SDK OpenTelemetry

Usa il `set_attribute` metodo per aggiungere attributi a ogni intervallo. Nota che per impostazione predefinita, tutti questi attributi span verranno convertiti in metadati nei dati grezzi di X-Ray. Per garantire che un attributo venga convertito in un'annotazione e non in metadati, puoi aggiungere la chiave degli attributi all'elenco degli attributi. `aws.xray.annotations` Per ulteriori informazioni, vedere Abilitare le annotazioni X-Ray personalizzate.

```
# 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
```

Strumentazione manuale Lambda

With X-Ray SDK

Dopo aver abilitato Active Tracing su Lambda, non sono necessarie configurazioni aggiuntive per utilizzare l'SDK X-Ray. Lambda creerà un segmento che rappresenta l'invocazione del gestore Lambda e potrai creare sottosegmenti o librerie di strumenti utilizzando l'SDK X-Ray senza alcuna configurazione aggiuntiva.

With OpenTelemetry SDK

Considerate il seguente codice di funzione Lambda di esempio (senza strumentazione):

```
require 'json'
def lambda_handler(event:, context:)
    # TODO implement
    { statusCode: 200, body: JSON.generate('Hello from Lambda!') }
end
```

Per strumentare manualmente la tua Lambda, dovrai:

1. Aggiungi le seguenti gemme per la tua 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. Inizializza l' OpenTelemetry SDK all'esterno del tuo Lambda Handler. Si consiglia di OpenTelemetry configurare l'SDK con:

   1. Un semplice processore Span con un X-Ray UDP Span Exporter per inviare Traces all'endpoint UDP X-Ray di Lambda
   2. Un propagatore Lambda a raggi X
   3. `service_name`configurazione da impostare sul nome della funzione Lambda

3. Nella tua classe Lambda Handler, aggiungi le seguenti righe allo strumento Lambda Handler:

```
  class Handler
        extend OpenTelemetry::Instrumentation::AwsLambda::Wrap
        ...

        instrument_handler :process
    end
```

Il codice seguente illustra la funzione Lambda dopo le modifiche richieste. È possibile creare intervalli personalizzati aggiuntivi per completare gli intervalli forniti 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
    )
  )

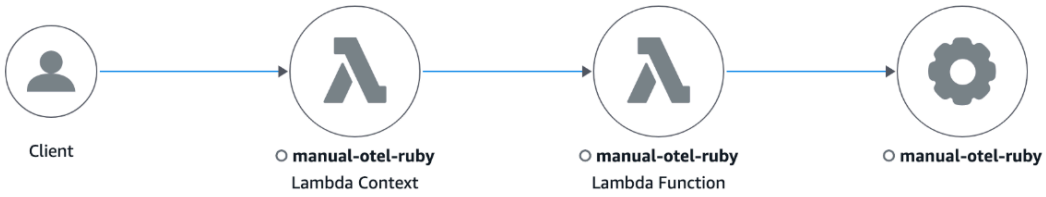
  # Configure X-Ray Lambda propagator
  c.propagators = [OpenTelemetry::Propagator::XRay.lambda_text_map_propagator]

  # 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
```

Di seguito è riportato un esempio di mappa di traccia di una funzione Lambda con strumentazione scritta in Ruby.



Puoi anche usare i layer Lambda OpenTelemetry per configurare la tua Lambda. Per ulteriori informazioni, consulta OpenTelemetry AWS-Lambda Instrumentation.

Creazione di risorse X-Ray con AWS CloudFormation

AWS X-Ray è integrato con AWS CloudFormation, un servizio che consente di modellare e configurare le AWS risorse in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. Crei un modello che descrive tutte le AWS risorse che desideri e fornisce CloudFormation e configura tali risorse per te.

Quando si utilizza CloudFormation, è possibile riutilizzare il modello per configurare le risorse X-Ray in modo coerente e ripetuto. Descrivi le tue risorse una sola volta, quindi fornisci le stesse risorse più e più volte in più regioni Account AWS .

X-Ray e modelli CloudFormation

Per fornire e configurare le risorse per X-Ray e i servizi correlati, è necessario conoscere CloudFormation i modelli. I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse che desideri fornire nei tuoi CloudFormation stack. Se non conosci JSON o YAML, puoi usare CloudFormation Designer per iniziare a usare i modelli. CloudFormation Per ulteriori informazioni, consulta Che cos'è CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation .

X-Ray supporta la creazione e `AWS::XRay::Group`l'utilizzo `AWS::XRay::SamplingRule`di `AWS::XRay::ResourcePolicy`risorse in. CloudFormationPer ulteriori informazioni, inclusi esempi di modelli JSON e YAML, consultate il riferimento ai tipi di risorse X-Ray nella Guida per l'utente.AWS CloudFormation

Scopri di più su CloudFormation

Per ulteriori informazioni CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- AWS CloudFormation Guida per l'utente
- CloudFormation Documentazione di riferimento delle API
- AWS CloudFormation Guida per l'utente dell'interfaccia a riga di comando

Etichettatura delle regole e dei gruppi di campionamento a raggi X

I tag sono parole o frasi che puoi usare per identificare e organizzare le tue risorse. AWS È possibile aggiungere più tag a ciascuna risorsa. Ogni tag include una chiave e un valore opzionale definiti dall'utente. Ad esempio, una chiave di tag potrebbe essere**`domain`**, e il valore del tag potrebbe essere**`example.com`**. Puoi cercare e filtrare le tue risorse in base ai tag che aggiungi. Per ulteriori informazioni sui modi di utilizzare i tag, consulta Etichettatura AWS delle risorse nella Guida AWS generale.

Puoi utilizzare i tag per applicare le autorizzazioni basate sui tag alle distribuzioni. CloudFront Per ulteriori informazioni, vedere Controllo dell'accesso alle AWS risorse utilizzando i tag delle risorse.

> **Note**
>
> Tag Editor e AWS Resource Groups attualmente non supportano le risorse X-Ray. Puoi aggiungere e gestire i tag utilizzando la AWS X-Ray console o l'API.

È possibile applicare tag alle risorse utilizzando la console X-Ray, l'API, AWS CLI SDKs, e. AWS Tools for Windows PowerShell Per ulteriori informazioni, consulta la seguente documentazione:

- API X-Ray: vedi le seguenti operazioni nel riferimento AWS X-Ray API:
  - ListTagsForResource
  - CreateSamplingRule
  - CreateGroup
  - TagResource
  - UntagResource
- AWS CLI — Vedi xray nel Command Reference AWS CLI
- SDKs — Consulta la documentazione SDK applicabile nella pagina Documentazione AWS

> **Note**
>
> Se non è possibile aggiungere o modificare tag su una risorsa X-Ray o non è possibile aggiungere una risorsa con tag specifici, è possibile che non si disponga delle autorizzazioni

necessarie per eseguire questa operazione. Per richiedere l'accesso, contattare un AWS utente dell'azienda che dispone delle autorizzazioni di amministratore in X-Ray.

Argomenti

- Limitazioni applicate ai tag
- Gestione dei tag nella console
- Gestione dei tag in AWS CLI
- Controlla l'accesso alle risorse X-Ray in base ai tag

Limitazioni applicate ai tag

Ai tag si applicano le limitazioni seguenti.

- Numero massimo di tag per risorsa: 50
- lunghezza massima della chiave: 128 caratteri Unicode;
- lunghezza massima del valore: 256 caratteri Unicode;
- Valori validi per la chiave e il valore - a-z, A-Z, 0-9, spazi e i seguenti caratteri: _ . : / = + - e @
- I valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole.
- Non utilizzare `aws:` come prefisso per le chiavi; è riservato all' AWS uso.

> **Note**
>
> Non è possibile modificare o eliminare i tag di sistema.

Gestione dei tag nella console

È possibile aggiungere tag opzionali durante la creazione di un gruppo X-Ray o di una regola di campionamento. I tag possono anche essere modificati o eliminati nella console in un secondo momento.

Le seguenti procedure spiegano come aggiungere, modificare ed eliminare i tag per i gruppi e le regole di campionamento nella console X-Ray.

Argomenti

- Aggiungi tag a un nuovo gruppo (console)
- Aggiungi tag a una nuova regola di campionamento (console)
- Modifica o elimina i tag per un gruppo (console)
- Modifica o elimina i tag per una regola di campionamento (console)

Aggiungi tag a un nuovo gruppo (console)

Quando si crea un nuovo gruppo X-Ray, è possibile aggiungere tag opzionali nella pagina Crea gruppo.

1. Accedi a Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Nel riquadro di navigazione, espandi Configurazione e scegli Gruppi.
3. Seleziona Crea gruppo.
4. Nella pagina Crea gruppo, specifica un nome e un'espressione di filtro per il gruppo. Per ulteriori informazioni su queste proprietà, consultare Configurazione dei gruppi.
5. In Tag, inserisci una chiave di tag e, facoltativamente, un valore di tag. Ad esempio, potete inserire una chiave di tag e un valore di **`Stage`** tag di**`Production`**, per indicare che questo gruppo è destinato all'uso in produzione. Quando aggiungete un tag, viene visualizzata una nuova riga in cui potete aggiungere un altro tag, se necessario. Per informazioni Limitazioni applicate ai tag sulle limitazioni relative ai tag, consulta questo argomento.
6. Quando hai finito di aggiungere i tag, scegli Crea gruppo.

Aggiungi tag a una nuova regola di campionamento (console)

Quando create una nuova regola di campionamento a raggi X, potete aggiungere tag nella pagina Crea regola di campionamento.

1. Accedi a Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Nel riquadro di navigazione, espandi Configurazione e scegli Sampling.
3. Scegliete Crea regola di campionamento.

4. Nella pagina Crea regola di campionamento, specifica un nome, una priorità, i limiti, i criteri di corrispondenza e gli attributi corrispondenti. Per ulteriori informazioni su queste proprietà, consultare Configurazione delle regole di campionamento di.
5. In Tag, inserisci una chiave di tag e, facoltativamente, un valore di tag. Ad esempio, potete inserire una chiave di tag e un valore di tag di **`StageProduction`**, per indicare che questa regola di campionamento è destinata all'uso in produzione. Quando aggiungete un tag, viene visualizzata una nuova riga in cui potete aggiungere un altro tag, se necessario. Per informazioni Limitazioni applicate ai tag sulle limitazioni relative ai tag, consulta questo argomento.
6. Quando hai finito di aggiungere i tag, scegli Crea regola di campionamento.

Modifica o elimina i tag per un gruppo (console)

È possibile modificare o eliminare i tag su un gruppo X-Ray nella pagina Modifica gruppo.

1. Accedi a Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Nel riquadro di navigazione, espandi Configurazione e scegli Gruppi.
3. Nella tabella Gruppi, scegli il nome di un gruppo.
4. Nella pagina Modifica gruppo, in Tag, modifica le chiavi e i valori dei tag. Non puoi avere chiavi di tag duplicate. I valori dei tag sono facoltativi; è possibile eliminare i valori se lo si desidera. Per ulteriori informazioni su altre proprietà nella pagina Modifica gruppo, vedereConfigurazione dei gruppi. Per informazioni Limitazioni applicate ai tag sulle limitazioni relative ai tag, consulta questo argomento.
5. Per eliminare un tag, scegli X a destra del tag.
6. Quando hai finito di modificare o eliminare i tag, scegli Aggiorna gruppo.

Modifica o elimina i tag per una regola di campionamento (console)

È possibile modificare o eliminare i tag su una regola di campionamento a raggi X nella pagina Modifica regola di campionamento.

1. Accedi a Console di gestione AWS e apri la console X-Ray a casahttps://console.aws.amazon.com/xray/.
2. Nel riquadro di navigazione, espandi Configurazione e scegli Sampling.
3. Nella tabella Regole di campionamento, scegliete il nome di una regola di campionamento.

4. In Tag, modificate le chiavi e i valori dei tag. Non puoi avere chiavi di tag duplicate. I valori dei tag sono facoltativi; è possibile eliminare i valori se lo si desidera. Per ulteriori informazioni su altre proprietà nella pagina Modifica regola di campionamento, vedereConfigurazione delle regole di campionamento di. Per informazioni Limitazioni applicate ai tag sulle limitazioni relative ai tag, consulta questo argomento.
5. Per eliminare un tag, scegli X a destra del tag.
6. Quando hai finito di modificare o eliminare i tag, scegli Aggiorna regola di campionamento.

Gestione dei tag in AWS CLI

È possibile aggiungere tag quando si crea un gruppo X-Ray o una regola di campionamento. È inoltre possibile utilizzare il AWS CLI per creare e gestire i tag. Per aggiornare i tag su un gruppo o una regola di campionamento esistente, usa la AWS X-Ray console o l'opzione TagResourceo UntagResource APIs.

Argomenti

- Aggiungere tag a un nuovo gruppo X-Ray o a una nuova regola di campionamento (CLI)
- Aggiungere tag a una risorsa esistente (CLI)
- Elenca i tag su una risorsa (CLI)
- Eliminare i tag su una risorsa (CLI)

Aggiungere tag a un nuovo gruppo X-Ray o a una nuova regola di campionamento (CLI)

Per aggiungere tag opzionali durante la creazione di un nuovo gruppo di raggi X o di una nuova regola di campionamento, utilizzate uno dei seguenti comandi.

- Per aggiungere tag a un nuovo gruppo, esegui il comando seguente, sostituendolo *group_name* con il nome del gruppo, *mydomain.com* con l'endpoint del servizio, *key_name* con una chiave di tag e, facoltativamente, *value* con un valore di tag. Per ulteriori informazioni su come creare un gruppo, consulta. Gruppi

```
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"}]
```

Di seguito è riportato un esempio di :

```
aws xray create-group \
   --group-name "AdminGroup" \
   --filter-expression "service(\"mydomain.com\") {fault OR error}" \
   --tags [{"Key": "Stage","Value": "Prod"},{"Key": "Department","Value": "QA"}]
```

- Per aggiungere tag a una nuova regola di campionamento, esegui il comando seguente, sostituendolo *key_name* con una chiave tag e, facoltativamente, *value* con un valore di tag. Questo comando specifica i valori del `--sampling-rule` parametro come file JSON. Per ulteriori informazioni su come creare una regola di campionamento, vedere. Regole di campionamento

```
aws xray create-sampling-rule \
   --cli-input-json file://file_name.json
```

Di seguito è riportato il contenuto del file *file_name.json* JSON specificato dal `--cli-input-json` parametro.

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

Il comando seguente è un esempio.

```
aws xray create-sampling-rule \
   --cli-input-json file://9000-base-scorekeep.json
```

Di seguito è riportato il contenuto del `9000-base-scorekeep.json` file di esempio specificato dal `--cli-input-json` parametro.

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

Aggiungere tag a una risorsa esistente (CLI)

È possibile eseguire il `tag-resource` comando per aggiungere tag a un gruppo X-Ray o a una regola di campionamento esistente. Questo metodo potrebbe essere più semplice dell'aggiunta di tag eseguendo o. `update-group` `update-sampling-rule`

Per aggiungere tag a un gruppo o a una regola di campionamento, esegui il comando seguente, sostituendo l'ARN con l'ARN della risorsa e specificando le chiavi e i valori opzionali dei tag che desideri aggiungere.

```
aws xray tag-resource \
   --resource-arn "ARN" \
   --tag-keys [{"Key":"key_name","Value":"value"}, {"Key":"key_name","Value":"value"}]
```

Di seguito è riportato un esempio di :

```
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"}]
```

Elenca i tag su una risorsa (CLI)

È possibile eseguire il `list-tags-for-resource` comando per elencare i tag di un gruppo X-Ray o di una regola di campionamento.

Per elencare i tag associati a un gruppo o a una regola di campionamento, esegui il comando seguente, sostituendo l'ARN con l'ARN della risorsa.

```
aws xray list-tags-for-resource \
   --resource-arn "ARN"
```

Di seguito è riportato un esempio di :

```
aws xray list-tags-for-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup"
```

Eliminare i tag su una risorsa (CLI)

È possibile eseguire il `untag-resource` comando per rimuovere i tag da un gruppo X-Ray o da una regola di campionamento.

Per rimuovere i tag da un gruppo o da una regola di campionamento, esegui il comando seguente, sostituendo l'ARN con l'ARN della risorsa e specificando le chiavi dei tag che desideri rimuovere.

È possibile rimuovere solo interi tag con il comando. `untag-resource` Per rimuovere i valori dei tag, usa la console X-Ray oppure elimina i tag e aggiungi nuovi tag con le stesse chiavi, ma valori diversi o vuoti.

```
aws xray untag-resource \
   --resource-arn "ARN" \
   --tag-keys ["key_name","key_name"]
```

Di seguito è riportato un esempio di :

```
aws xray untag-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/group_name" \
   --tag-keys ["Stage","Department"]
```

Controlla l'accesso alle risorse X-Ray in base ai tag

È possibile allegare tag ai gruppi di raggi X o alle regole di campionamento oppure passare i tag in una richiesta a X-Ray. Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `xray:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Per saperne di più su queste chiavi di condizione, consulta Controllo dell'accesso alle AWS risorse utilizzando i tag delle risorse.

Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta Gestione dell'accesso ai gruppi X-Ray e alle regole di campionamento basate sui tag.

Risoluzione dei problemi AWS X-Ray

Questo argomento elenca gli errori e i problemi più comuni che potresti riscontrare quando utilizzi l'API X-Ray, la console o. SDKs Se scopri un problema che non è elencato qui di seguito, puoi utilizzare il pulsante Feedback in questa pagina per segnalarlo.

Sections

- Mappa di tracciamento a raggi X e pagine con i dettagli della traccia
- X-Ray SDK per Java
- X-Ray SDK per Node.js
- Il demone X-Ray

Mappa di tracciamento a raggi X e pagine con i dettagli della traccia

Le seguenti sezioni possono essere utili in caso di problemi con l'utilizzo della mappa di tracciamento a raggi X e della pagina dei dettagli della traccia:

Non vedo tutti i miei registri CloudWatch

La modalità di configurazione dei log in modo che vengano visualizzati nella mappa di tracciamento a raggi X e nelle pagine dei dettagli della traccia dipende dal servizio.

- I log del gateway API vengono visualizzati se la registrazione è attivata in API Gateway.

Non tutti i nodi della mappa dei servizi supportano la visualizzazione dei log associati. Visualizza i log per i seguenti tipi di nodi:

- Contesto Lambda
- funzione Lambda
- Fase API Gateway
- Cluster Amazon ECS
- Istanza Amazon ECS
- Servizio Amazon ECS
- Attività di Amazon ECS

- Cluster Amazon EKS
- Spazio dei nomi Amazon EKS
- Nodo Amazon EKS
- Pod Amazon EKS
- Servizio Amazon EKS

Non vedo tutti i miei allarmi sulla mappa di tracciamento a raggi X

La mappa di tracciamento a raggi X mostra solo l'icona di avviso per un nodo se gli allarmi associati a quel nodo sono nello stato ALARM.

La mappa di traccia associa gli allarmi ai nodi utilizzando la seguente logica:

- Se il nodo rappresenta un AWS servizio, tutti gli allarmi con lo spazio dei nomi associato a quel servizio sono associati al nodo. Ad esempio, un nodo di tipo C `AWS::Kinesis` è collegato a tutti gli allarmi basati sulle metriche del namespace. CloudWatch `AWS/Kinesis`
- Se il nodo rappresenta una AWS risorsa, gli allarmi su quella risorsa specifica sono collegati. Ad esempio, un nodo di tipo `AWS::DynamoDB::Table` con il nome «MyTable» è collegato a tutti gli allarmi basati su una metrica con lo spazio dei nomi `AWS/DynamoDB` e con la dimensione impostata su. `TableName` `MyTable`
- Se il nodo è di tipo sconosciuto, identificato da un bordo tratteggiato attorno al nome, a tale nodo non vengono associati allarmi.

Non vedo alcune AWS risorse sulla mappa di tracciamento

Non tutte le AWS risorse sono rappresentate da un nodo dedicato. Alcuni AWS servizi sono rappresentati da un singolo nodo per tutte le richieste al servizio. I seguenti tipi di risorse vengono visualizzati con un nodo per risorsa:

- `AWS::DynamoDB::Table`
- `AWS::Lambda::Function`

  Le funzioni Lambda sono rappresentate da due nodi, uno per il contenitore Lambda e uno per la funzione. Questo aiuta a identificare i problemi di avvio a freddo con le funzioni Lambda. I nodi del container Lambda sono associati ad allarmi e pannelli di controllo allo stesso modo dei nodi funzione Lambda.

- `AWS::ApiGateway::Stage`
- `AWS::SQS::Queue`
- `AWS::SNS::Topic`

Ci sono troppi nodi sulla mappa di traccia

Utilizza i gruppi X-Ray per suddividere la mappa in più mappe. Per ulteriori informazioni, consulta Utilizzo delle espressioni filtro con i gruppi.

X-Ray SDK per Java

Errore: eccezione nel thread «Thread-1" com.amazonaws.xray.exceptions. SegmentNotFoundException: Impossibile iniziare il sottosegmento denominato 'AmazonSNS': il segmento non può essere trovato.

Questo errore indica che X-Ray SDK ha tentato di registrare una chiamata in uscita verso AWS, ma non è riuscito a trovare un segmento aperto. Ciò può verificarsi nelle seguenti situazioni:

- Un filtro servlet non è configurato: l'SDK X-Ray crea segmenti per le richieste in entrata con un filtro denominato. `AWSXRayServletFilter` Configura un filtro servlet per analizzare le richieste in entrata.
- Stai utilizzando client strumentati al di fuori del codice servlet: se utilizzi un client con strumenti per effettuare chiamate nel codice di avvio o in altro codice che non viene eseguito in risposta a una richiesta in arrivo, devi creare un segmento manualmente. Per esempi, consulta Analisi del codice di avvio.
- Stai usando client strumentati nei thread di lavoro: quando crei un nuovo thread, il registratore X-Ray perde il riferimento al segmento aperto. È possibile utilizzare i `setTraceEntity`metodi `getTraceEntity`and per ottenere un riferimento al segmento o al sottosegmento corrente (`Entity`) e passarlo nuovamente al registratore all'interno del thread. Consulta Utilizzo dei client analizzati nei thread worker per un esempio.

X-Ray SDK per Node.js

Problema: CLS non funziona con Sequelize

Passa lo spazio dei nomi X-Ray SDK per Node.js a Sequelize con il metodo. `cls`

```
var AWSXRay = require('aws-xray-sdk');
const Sequelize = require('sequelize');
Sequelize.cls = AWSXRay.getNamespace();
const sequelize = new Sequelize(...);
```

Problema: CLS non funziona con Bluebird

Utilizza `cls-bluebird` per fare in modo che Bluebird funzioni con CLS.

```
var AWSXRay = require('aws-xray-sdk');
var Promise = require('bluebird');
var clsBluebird = require('cls-bluebird');
clsBluebird(AWSXRay.getNamespace());
```

Il demone X-Ray

Problema: il daemon sta utilizzando delle credenziali errate

Il demone utilizza l'SDK per caricare le AWS credenziali. Se si utilizzano più metodi di fornitura delle credenziali, viene utilizzato il metodo con la massima precedenza. Per ulteriori informazioni, consulta Esecuzione del daemon.

Cronologia dei documenti per AWS X-Ray

La tabella seguente descrive le modifiche importanti alla documentazione per. AWS X-Ray Per ricevere notifiche sugli aggiornamenti di questa documentazione, è possibile iscriversi a un feed RSS.

Ultimo aggiornamento della documentazione: 7 marzo 2024

Modifica	Descrizione	Data
Cronologia aggiornata del AWS X-Ray supporto per SDK e Daemon	Il 25 febbraio 2026, AWS X-Ray SDKs/Daemon entrerà in modalità manutenzione, in cui AWS limiterà le versioni di X-Ray SDK e Daemon per risolvere solo i problemi di sicurezza. Per ulteriori informazioni sulla tempistica del supporto, consulta la cronologia di supporto di X-Ray SDK e Daemon. Ti consigliamo di migrare a. OpenTelemetry Per ulteriori informazioni sulla migrazione a OpenTelemetry, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry	26 novembre 2025
È stato aggiunto un avviso di fine supporto per SDK e Daemon AWS X-Ray	Il 25 febbraio 2027, AWS X-Ray interromperà il supporto per SDK e Daemon. AWS X-Ray Ti consigliamo di eseguire la migrazione a. OpenTelemetry Per ulteriori informazioni, vedere Migrazione dalla X-Ray strumentazione alla	22 agosto 2025

	strumentazione. OpenTelemetry	
Funzionalità aggiunta	Migrazione da a. X-Ray OpenTelemetry Per ulteriori informazioni, vedere Migrazione dalla X-Ray strumentazione alla strumentazione. OpenTelemetry	13 giugno 2025
Funzionalità aggiunta	AWS X-Ray ora supporta la ricerca delle transazioni. Per ulteriori informazioni, consulta Transaction Search.	21 novembre 2024
Funzionalità aggiunta	AWS X-Ray ora supporta OpenTelemetry Protocol (OTLP) Endpoint. Per ulteriori informazioni, consulta OpenTelemetry.	21 novembre 2024
Funzionalità aggiunta	X-Ray ora registra gli eventi relativi ai dati`PutTraceSegments` , tra cui`GetTraceSummaries` , e `BatchGetTraces` a AWS CloudTrail. X-Ray ora registra anche l'evento `GetSamplingStatisticSummaries` di gestione su CloudTrail. Per ulteriori informazioni, consulta Registrazione delle chiamate X-Ray API con AWS CloudTrail.	7 marzo 2024

Funzionalità aggiunta	X-Ray ora supporta gli ID di traccia creati tramite OpenTelemetry o qualsiasi altro framework conforme alla specifica W3C Trace Context. Per ulteriori informazioni, vedere Invio di dati di traccia a. X-Ray	25 ottobre 2023
Funzionalità aggiunta	Ora puoi configurare il tracciamento attivo di Amazon SNS, consentendoti di tracciare e analizzare le richieste man mano che attraversano i tuoi argomenti di Amazon SNS. Per ulteriori informazioni, consulta Amazon SNS e. AWS X-Ray	8 febbraio 2023
X-Ray SDK aggiornato per argomento Node.js	Sono stati aggiunti dettagli per la strumentazione dei client che utilizzano la V3. AWS SDK per JavaScript Per i dettagli, consulta Tracciare le chiamate SDK con l' AWS SDK per. X-Ray Node.js	7 febbraio 2023

Dettagli aggiornati sulla policy gestita da IAM	È stata aggiunta l'autorizzazione IAM per l'osservabilità tra account alle policy `AWSXrayCrossAccountSharingConfiguration` gestite `AWSXRayFullAccess` e a `AWSXRayReadOnlyAccess` quelle gestite. Per i dettagli, consulta le politiche gestite da IAM per. X-Ray	7 febbraio 2023
Funzionalità aggiunta	AWS X-Ray ora supporta l'osservabilità tra più account, consentendo di monitorare e risolvere i problemi delle applicazioni che si estendono su più account all'interno di un. Regione AWS Per i dettagli, vedi tracing. Cross-account	27 novembre 2022
Funzionalità aggiunta	Ora puoi visualizzare le tracce collegate tra i produttori di messaggi, una coda Amazon SQS e i consumatori, fornendo una visione connessa delle tracce inviate da applicazioni basate sugli eventi. Per ulteriori informazioni, consulta Tracing event-driven applications. https://docs.aws.amazon.com/xray/latest/devguide/xray-tracelinking.html	20 novembre 2022

Dettagli aggiornati sulla policy gestita da IAM	È stata aggiunta l'autorizzazione IAM per elencare le politiche relative alle risorse nella policy `AWSXRayReadOnlyAccess` gestita. Per i dettagli, consulta le politiche gestite da IAM per X-Ray.	15 novembre 2022
Autorizzazioni della console IAM aggiornate e dettagli sulle policy gestite	Il set di autorizzazioni IAM utilizzato dalla X-Ray console è stato aggiornato, insieme alla descrizione della policy `AWSXRayReadOnlyAccess` gestita. Per i dettagli, consulta Utilizzo della X-Ray console.	11 novembre 2022
È stata aggiunta AWS Distro for Ruby OpenTelemetry	AWS Distro for OpenTelemetry (ADOT) fornisce un unico set di API, librerie e agenti open source per raccogliere tracce e metriche distribuite. ADOT Ruby ti consente di strumentare la tua applicazione Ruby for e altri back-end di tracciamento. X-Ray Per ulteriori informazioni, consulta Distro for Ruby. AWS OpenTelemetry	7 febbraio 2022
Funzionalità aggiunta	Ora puoi visualizzare le tracce e configurarle X-Ray dalla CloudWatch console. Per ulteriori informazioni, vedere X-Ray console.	24 gennaio 2022

CloudWatch RUM integrato	Con AWS X-Ray and CloudWatch RUM, puoi analizzare ed eseguire il debug del percorso delle richieste a partire dagli utenti finali della tua applicazione attraverso i servizi AWS gestiti a valle. Per ulteriori informazioni, consulta CloudWatch RUM e. AWS X-Ray	3 dicembre 2021
AWS Distro integrato per OpenTelemetry	AWS Distro for OpenTelemetry (ADOT) fornisce un unico set di API, librerie e agenti open source per raccogliere tracce e metriche distribuite. ADOT consente di strumentare l'applicazione e altri back-end di tracciamento X-Ray . Per ulteriori informazioni, consulta Strumenting your app.	23 settembre 2021
Funzionalità aggiunta	AWS X-Ray ora si integra con Amazon Virtual Private Cloud, consentendo alle risorse di Amazon VPC di comunicare con il X-Ray servizio senza passare attraverso la rete Internet pubblica. Per ulteriori informazioni, consulta Utilizzo AWS X-Ray con endpoint VPC.	20 maggio 2021

Funzionalità aggiunta	AWS X-Ray ora si integra con CloudFormation, consentendoti di effettuare il provisioning e configurare le risorse. X-Ray Per ulteriori informazioni, consulta Creazione di X-Ray risorse con CloudFormation.	06 maggio 2021
Funzionalità aggiunta	AWS X-Ray ora si integra con Amazon EventBridge per tracciare gli eventi trasmessi EventBridge. Ciò fornisce agli utenti una visione più completa del proprio sistema. Per ulteriori informazioni, consulta Amazon EventBridge e AWS X-Ray.	2 marzo 2021
È stato aggiunto un demone a ECR	Il demone può ora essere scaricato da Amazon ECR. Per ulteriori informazioni, consulta Download del daemon.	1° marzo 2021
Funzionalità aggiunta	AWS X-Ray ora supporta le notifiche relative agli approfondimenti ad Amazon EventBridge. Ciò ti consente di intraprendere azioni automatiche sugli approfondimenti utilizzando EventBridge. Per ulteriori informazioni, consulta Notifiche di Insights.	15 ottobre 2020

Demoni scaricabili aggiunti	AWS X-Ray introduce il demone di supporto per Linux ARM64. Per ulteriori informazioni, vedere daemonbrazil ws AWS X-Ray	1° ottobre 2020
Funzionalità aggiunta	AWS X-Ray ora supporta l'integrazione attiva con Amazon Synthetics. CloudWatch Ciò consente di visualizzare dettagli su un nodo client Synthetics Canary, ad esempio il tempo di risposta e lo stato. Puoi anche eseguire analisi nella console Analytics in base alle informazioni provenienti da un nodo client Synthetics Canary. Per ulteriori informazioni, consulta Debugging CloudWatch synthetics canaries using. X-Ray	24 settembre 2020
Funzionalità aggiunta	AWS X-Ray ora supporta la tracciabilità dei flussi di lavoro end-to-end per. AWS Step Functions Puoi visualizzare i componenti della tua macchina a stati, identificare i colli di bottiglia nelle prestazioni e risolvere le richieste che hanno provocato un errore. Per ulteriori informazioni, vedere e. AWS Step Functions AWS X-Ray	14 settembre 2020

Funzionalità aggiunta	AWS X-Ray introduce approfondimenti per analizzare continuamente i dati di tracciamento nel tuo account per identificare i problemi emergenti nelle tue applicazioni. Insights registra gli incidenti e ne traccia l'impatto fino alla risoluzione. Per ulteriori informazioni, consulta Using Insights in the console AWS X-Ray	3 settembre 2020
Funzionalità aggiunta	AWS X-Ray introduce l'agente di strumentazione automatica a Java, che consente ai clienti di raccogliere dati di tracciamento senza dover modificare l'applicazione esistente. Java-based È ora possibile tracciare le applicazioni Java web e basate su servlet con modifiche minime alla configurazione e senza modifiche al codice. Per ulteriori informazioni, vedere AWS X-Ray Auto Instrumentation Agent for Java.	3 settembre 2020

Funzionalità aggiunta	AWS X-Ray ha aggiunto una nuova pagina Gruppi alla X-Ray console per facilitare la creazione e la gestione di gruppi di tracce. Per ulteriori informazioni, vedere Configurazione dei gruppi nella X-Ray console.	24 agosto 2020
Funzionalità aggiunta	AWS X-Ray ora consente di aggiungere tag ai gruppi e alle regole di campionamento. Puoi anche controllare l'accesso ai gruppi e alle regole di campionamento in base ai tag. Per ulteriori informazioni, consulta Assegnazione di tag a regole e gruppi di X-Ray campionamento e Gestione dell'accesso ai X-Ray gruppi e alle regole di campionamento basate sui tag.	24 agosto 2020

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.