



Panduan Developer

AWS X-Ray



AWS X-Ray: Panduan Developer

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang mungkin menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan properti dari masing-masing pemilik, yang mungkin berafiliasi, terkait dengan, atau disponsori oleh Amazon, atau tidak.

Table of Contents

Apa itu AWS X-Ray?	1
Memulai	4
Memilih antarmuka	6
Menggunakan SDK	7
Gunakan ADOT SDK	8
Gunakan X-Ray SDK	10
Gunakan konsol	11
Gunakan CloudWatch konsol Amazon	12
Gunakan konsol X-Ray	12
Jelajahi konsol X-Ray	13
Peta jejak	14
Pelacakan	22
Ekspresi Filter	31
Penelusuran lintas akun	43
Menelusuri aplikasi berbasis peristiwa	47
Histogram	51
Wawasan	53
Analitik	62
Grup	69
Pengambilan sampel	79
Konsol deep linking	85
Gunakan API X-Ray	88
Tutorial	90
Mengirim data	95
Mendapatkan data	100
Konfigurasi	114
Pengambilan sampel	122
Dokumen segmen	126
Konsep	146
Segmen	146
Subsegmen	147
Grafik layanan	151
Pelacakan	152
Pengambilan sampel	153

Header penelusuran	154
Ekspresi filter	155
Grup	156
Anotasi dan metadata	156
Kesalahan, cacat, dan pengecualian	157
Keamanan	158
.....	158
Perlindungan data	158
Manajemen identitas dan akses	161
Audiens	161
Mengautentikasi dengan identitas	162
Mengelola akses menggunakan kebijakan	165
Bagaimana AWS X-Ray bekerja dengan IAM	168
Contoh kebijakan berbasis identitas	177
Pemecahan Masalah	190
Pencatatan dan pemantauan	192
Validasi kepatuhan	193
Ketahanan	194
Keamanan infrastruktur	195
Titik akhir VPC	195
Membuat VPC endpoint untuk X-Ray	196
Mengendalikan akses ke X-Ray VPC Endpoint Anda	197
Wilayah yang Didukung	198
Aplikasi sampel	200
Tutorial Scorekeep	202
Prasyarat	203
Instal aplikasi Scorekeep menggunakan CloudFormation	204
Hasilkan data pelacakan	205
Lihat peta jejak di AWS Management Console	206
Mengonfigurasi notifikasi Amazon SNS	214
Jelajahi aplikasi sampel	215
Opsional: Kebijakan hak istimewa terbatas	220
Bersihkan	223
Langkah selanjutnya	224
AWS Klien SDK	224
Subsegmen kustom	225

Anotasi dan metadata	225
Klien HTTP	227
Klien SQL	227
AWS Lambda fungsi	230
Nama acak	231
Pekerja	233
Menginstrumentasi kode perusahaan rintisan	235
Skrip instrumentasi	238
Menginstrumentasi klien web	239
Utas pekerja	243
X-Ray daemon	245
Mengunduh daemon	245
Memverifikasi tanda tangan arsip daemon	247
Menjalankan daemon	248
Memberikan izin kepada daemon untuk mengirim data ke X-Ray	249
Log daemon X-Ray	249
Konfigurasi	250
Variabel lingkungan yang didukung	251
Menggunakan opsi baris perintah	251
Menggunakan file konfigurasi	252
Jalankan daemon secara lokal	254
Menjalankan daemon X-Ray di Linux	254
Menjalankan daemon X-Ray di kontainer Docker	254
Menjalankan Daemon X-Ray di Windows	256
Menjalankan daemon X-Ray di OS X	257
Pada Elastic Beanstalk	257
Menggunakan integrasi X-Ray Elastic Beanstalk untuk menjalankan X-Ray daemon	258
Mengunduh dan menjalankan daemon X-Ray secara manual (lanjutan)	259
Di Amazon EC2	261
Di Amazon ECS	263
Menggunakan citra Docker resmi	263
Buat dan bangun citra Docker	263
Konfigurasikan opsi baris perintah di konsol Amazon ECS	267
Integrasi dengan Layanan AWS	268
AWS Distro untuk OpenTelemetry	270
AWS Distro untuk OpenTelemetry	270

API Gateway	271
App Mesh	273
App Runner	276
AWS AppSync	276
CloudTrail	276
Acara manajemen X-Ray di CloudTrail	278
Peristiwa data X-Ray di CloudTrail	279
Contoh acara X-Ray	280
CloudWatch	283
CloudWatch RUM	284
CloudWatch Synthetics	285
AWS Config	294
Membuat pemicu fungsi Lambda	295
Membuat AWS Config aturan khusus untuk x-ray	296
Contoh hasil	297
Notifikasi Amazon SNS	298
Amazon EC2	298
Elastic Beanstalk	298
Penyeimbang Beban Elastis	299
EventBridge	299
Melihat sumber dan target pada peta layanan X-Ray	300
Sebarkan konteks penelusuran untuk target peristiwa	300
Lambda	306
Amazon SNS	308
Konfigurasi penelusuran aktif Amazon SNS	309
Lihat jejak penerbit dan pelanggan Amazon SNS di konsol X-Ray	310
Step Functions	312
Amazon SQS	313
Kirim header pelacakan HTTP	315
Mengambil header pelacakan dan memulihkan konteks pelacakan	315
Amazon S3	316
Konfigurasi notifikasi Peristiwa Amazon S3	317
Instrumentasi aplikasi Anda	318
Menginstrumentasi aplikasi Anda dengan AWS Distro untuk OpenTelemetry	318
Menginstrumentasi aplikasi Anda dengan AWS X-Ray SDKs	320
Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs	321

Pencarian Transaksi	323
OpenTelemetry Titik Akhir Protokol (OTLP)	324
Bekerja dengan Go	325
AWS Distro untuk Go OpenTelemetry	325
X-Ray SDK for Go	325
Persyaratan	327
Referensi dokumentasi	327
Konfigurasi	328
Permintaan masuk	334
AWS Klien SDK	337
Panggilan HTTP keluar	339
Kueri SQL	340
Subsegmen kustom	340
Anotasi dan metadata	341
Bekerja dengan Java	344
AWS Distro untuk OpenTelemetry Java	344
X-Ray SDK for Java	344
Submodul	346
Persyaratan	347
Manajemen dependensi	348
Agen instrumentasi otomatis	349
Konfigurasi	363
Permintaan masuk	374
AWS Klien SDK	379
Panggilan HTTP keluar	381
Kueri SQL	383
Subsegmen kustom	386
Anotasi dan metadata	388
Pemantauan	393
Multithreading	397
AOP dengan Spring	398
Bekerja menggunakan Node.js	404
AWS Distro untuk OpenTelemetry JavaScript	404
X-Ray SDK untuk Node.js	405
Persyaratan	407
Manajemen dependensi	407

Sampel Node.js	408
Konfigurasi	408
Permintaan masuk	414
AWS Klien SDK	417
Panggilan HTTP keluar	421
Kueri SQL	423
Subsegmen kustom	424
Anotasi dan metadata	426
Bekerja menggunakan Python	431
AWS Distro untuk Python OpenTelemetry	431
SDK X-Ray untuk Python	431
Persyaratan	434
Manajemen dependensi	434
Konfigurasi	435
Permintaan masuk	442
Pustaka patching	448
AWS Klien SDK	451
Panggilan HTTP keluar	452
Subsegmen kustom	453
Anotasi dan metadata	455
Instrumen aplikasi nirserver	459
Bekerja dengan .NET	466
AWS Distro para OpenTelemetry .NET	466
X-Ray SDK for .NET	466
Persyaratan	469
Menambahkan X-Ray SDK for .NET untuk aplikasi Anda	469
Manajemen dependensi	469
Konfigurasi	471
Permintaan masuk	478
AWS Klien SDK	482
Panggilan HTTP keluar	484
Kueri SQL	486
Subsegmen kustom	489
Anotasi dan Metadata	490
Bekerja dengan Ruby	494
AWS Distro para Ruby OpenTelemetry	494

X-Ray SDK for Ruby	494
Persyaratan	496
Konfigurasi	496
Permintaan masuk	503
Pustaka patching	507
AWS Klien SDK	507
Subsegmen kustom	509
Anotasi dan metadata	510
Migrasi dari instrumentasi X-Ray ke instrumentasi OpenTelemetry	514
Memahami OpenTelemetry	514
OpenTelemetry dukungan di AWS	515
Memahami OpenTelemetry konsep migrasi	516
Membandingkan fitur	517
Menyiapkan dan mengonfigurasi penelusuran	518
Mendeteksi sumber daya di lingkungan Anda	519
Mengelola strategi pengambilan sampel	519
Mengelola konteks jejak	520
Menyebarkan konteks jejak	521
Menggunakan instrumentasi perpustakaan	521
Mengekspor jejak	522
Memproses dan meneruskan jejak	522
Pemrosesan rentang (OpenTelemetry-konsep spesifik)	523
Bagasi (konsep OpenTelemetry -soecific)	523
Ikhtisar migrasi	524
Rekomendasi untuk aplikasi baru dan yang sudah ada	524
Menelusuri perubahan pengaturan	525
Perubahan instrumentasi perpustakaan	525
Perubahan instrumentasi lingkungan Lambda	526
Membuat data jejak secara manual	526
Migrasi dari X-Ray Daemon ke AWS CloudWatch agen atau kolektor OpenTelemetry	527
Migrasi di Amazon EC2 atau server lokal	527
Migrasi di Amazon ECS	531
Migrasi pada Elastic Beanstalk	535
Migrasi ke OpenTelemetry Java	536
Solusi instrumentasi otomatis kode nol	537
Solusi instrumentasi manual dengan SDK	537

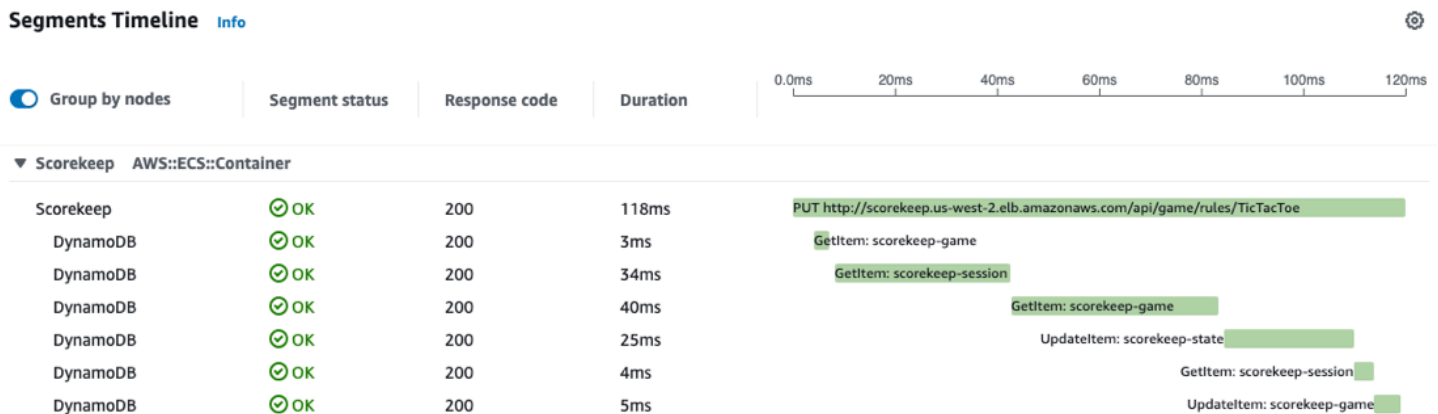
Menelusuri permintaan yang masuk (instrumentasi kerangka pegas)	540
AWS Instrumentasi SDK v2	541
Instrumentasi panggilan HTTP keluar	543
Dukungan instrumentasi untuk perpustakaan lain	544
Membuat data jejak secara manual	544
Instrumentasi Lambda	547
Migrasi ke Go OpenTelemetry	553
Instrumentasi manual dengan SDK	553
Menelusuri permintaan yang masuk (instrumentasi penanganan HTTP)	555
AWS Instrumentasi SDK for Go v2	556
Instrumentasi panggilan HTTP keluar	557
Dukungan instrumentasi untuk perpustakaan lain	559
Membuat data jejak secara manual	559
Instrumentasi manual Lambda	561
Migrasi ke Node.js OpenTelemetry	567
Solusi instrumentasi otomatis kode nol	568
Solusi instrumentasi manual	568
Menelusuri permintaan yang masuk	572
AWS Instrumentasi SDK JavaScript V3	556
Instrumentasi panggilan HTTP keluar	575
Dukungan instrumentasi untuk perpustakaan lain	576
Membuat data jejak secara manual	559
Instrumentasi Lambda	561
Migrasi ke OpenTelemetry .NET	579
Solusi instrumentasi otomatis kode nol	580
Solusi instrumentasi manual dengan SDK	581
Membuat data jejak secara manual	584
Menelusuri permintaan yang masuk (instrumentasi inti ASP.NET dan ASP.NET)	587
AWS Instrumentasi SDK	588
Instrumentasi panggilan HTTP keluar	589
Dukungan instrumentasi untuk perpustakaan lain	589
Instrumentasi Lambda	561
Migrasi ke Python OpenTelemetry	594
Solusi instrumentasi otomatis kode nol	595
Instrumen aplikasi Anda secara manual	595
Menelusuri inisialisasi pengaturan	596

Menelusuri permintaan yang masuk	598
AWS Instrumentasi SDK	600
Menginstrumentasi panggilan HTTP keluar melalui permintaan	602
Dukungan instrumentasi untuk perpustakaan lain	603
Membuat data jejak secara manual	603
Instrumentasi Lambda	605
Migrasi ke Ruby OpenTelemetry	606
Instrumentasikan solusi Anda secara manual dengan SDK	607
Menelusuri permintaan masuk (instrumentasi Rails)	609
AWS Instrumentasi SDK	610
Instrumentasi panggilan HTTP keluar	611
Dukungan instrumentasi untuk perpustakaan lain	612
Membuat data jejak secara manual	612
Instrumentasi manual Lambda	615
Membuat sumber daya X-Ray dengan CloudFormation	618
X-Ray dan AWS CloudFormation template	618
Pelajari lebih lanjut tentang AWS CloudFormation	618
Penandaan	619
Batasan tag	620
Mengelola tanda di konsol tersebut	620
Tambahkan tanda ke grup baru (konsol)	621
Tambahkan tanda ke aturan pengambilan sampel baru (konsol)	621
Edit atau hapus tanda untuk grup (konsol)	622
Edit atau hapus tanda untuk aturan pengambilan sampel (konsol)	622
Mengelola tag di AWS CLI	623
Tambahkan tanda ke grup X-Ray atau aturan pengambilan sampel baru (CLI)	623
Tambahkan tanda ke sumber daya yang ada (CLI)	625
Daftar tanda pada sumber daya (CLI)	626
Hapus tanda pada sumber daya (CLI)	626
Kontrol akses ke sumber daya X-Ray berdasarkan tanda	627
Pemecahan Masalah	628
Peta jejak X-Ray dan halaman detail jejak	628
Saya tidak melihat semua CloudWatch log saya	628
Saya tidak melihat semua alarm saya di peta jejak X-Ray	629
Saya tidak melihat beberapa AWS sumber daya di peta jejak	629
Ada terlalu banyak node di peta jejak	630

X-Ray SDK for Java	630
X-Ray SDK untuk Node.js	630
X-Ray Daemon	631
Riwayat Dokumen	632
.....	dcxli

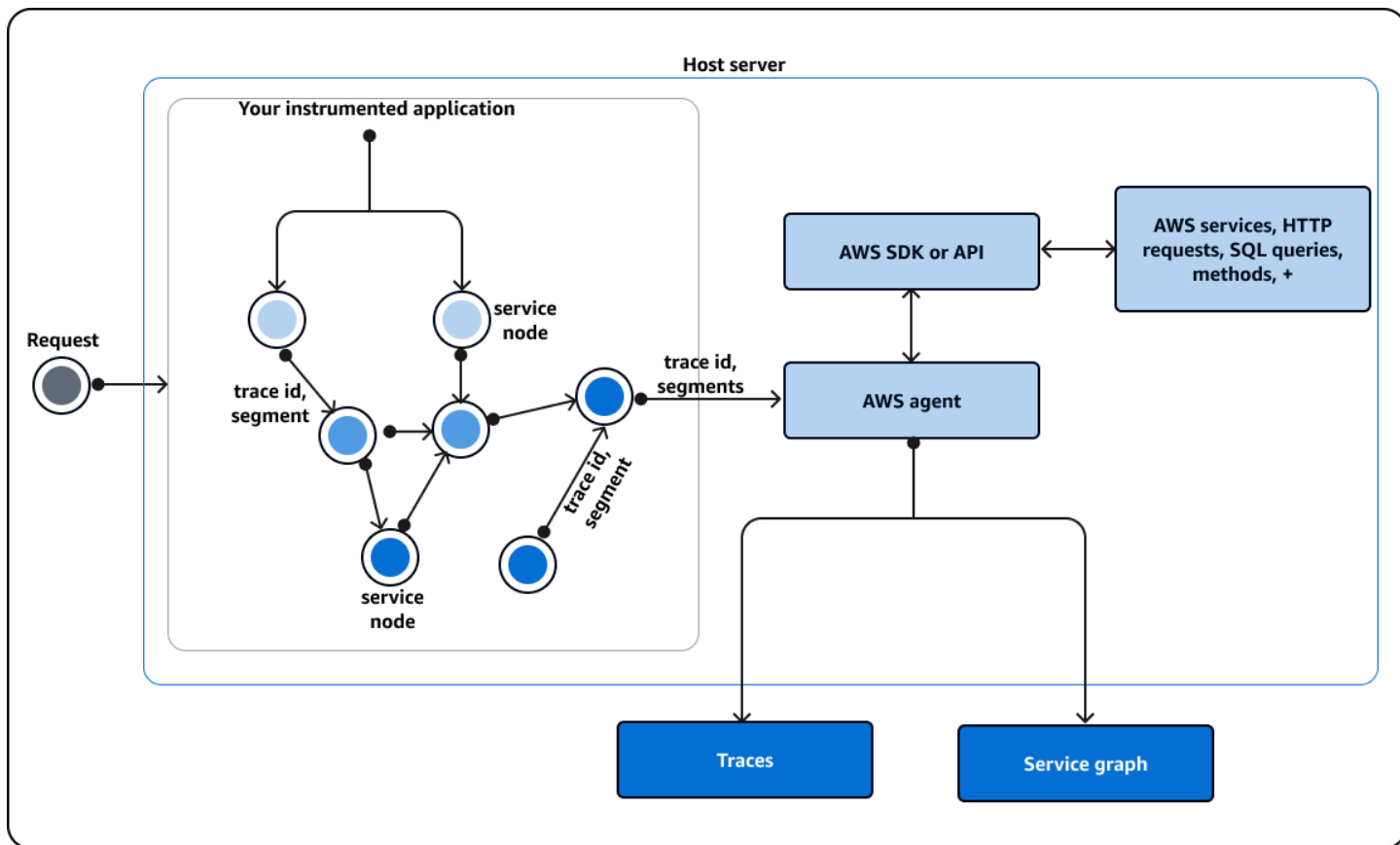
Apa itu AWS X-Ray?

AWS X-Ray adalah layanan yang mengumpulkan data tentang permintaan yang disajikan aplikasi Anda, dan menyediakan alat yang dapat Anda gunakan untuk melihat, memfilter, dan mendapatkan wawasan tentang data tersebut guna mengidentifikasi masalah dan peluang pengoptimalan. Untuk setiap permintaan yang dilacak ke aplikasi Anda, Anda dapat melihat informasi terperinci tidak hanya tentang permintaan dan respons, tetapi juga tentang panggilan yang dilakukan aplikasi Anda ke AWS sumber daya hilir, layanan mikro, database, dan web. APIs



AWS X-Ray menerima jejak dari aplikasi Anda, selain penggunaan aplikasi Layanan AWS Anda yang sudah terintegrasi dengan X-Ray. Instrumentasi aplikasi Anda melibatkan pengiriman data jejak untuk permintaan masuk dan keluar dan peristiwa lain dalam aplikasi Anda, bersama dengan metadata tentang setiap permintaan. Banyak skenario instrumentasi hanya memerlukan perubahan konfigurasi. Misalnya, Anda dapat menginstruksikan semua permintaan HTTP masuk dan panggilan hilir ke Layanan AWS aplikasi Java Anda. Ada beberapa agen SDKs, dan alat yang dapat digunakan untuk instrumen aplikasi Anda untuk penelusuran X-Ray. Lihat [Instrumentasi aplikasi Anda](#) untuk informasi selengkapnya.

Layanan AWS yang [terintegrasi dengan X-Ray](#) dapat menambahkan header penelusuran ke permintaan yang masuk, mengirim data jejak ke X-Ray, atau menjalankan daemon X-Ray. Misalnya, AWS Lambda dapat mengirim data jejak tentang permintaan ke fungsi Lambda Anda, dan menjalankan daemon X-Ray pada pekerja agar lebih mudah menggunakan X-Ray SDK.



Alih-alih mengirim data jejak langsung ke X-Ray, setiap SDK klien mengirimkan dokumen segmen JSON ke proses daemon yang mendengarkan lalu lintas UDP. [X-Ray Daemon](#) menyangga segmen dalam antrean dan mengunggah segmen tersebut ke X-Ray dalam batch. Daemon tersedia untuk Linux, Windows, dan macOS, dan disertakan pada dan platform. AWS Elastic Beanstalk AWS Lambda

X-Ray menggunakan data jejak dari AWS sumber daya yang memberi daya pada aplikasi cloud Anda untuk menghasilkan peta jejak terperinci. Peta jejak menunjukkan klien, layanan front-end Anda, dan layanan backend yang dipanggil layanan front-end Anda untuk memproses permintaan dan mempertahankan data. Gunakan peta jejak untuk mengidentifikasi kemacetan, lonjakan latensi, dan masalah lain yang harus diselesaikan guna meningkatkan kinerja aplikasi Anda.



Memulai dengan X-Ray

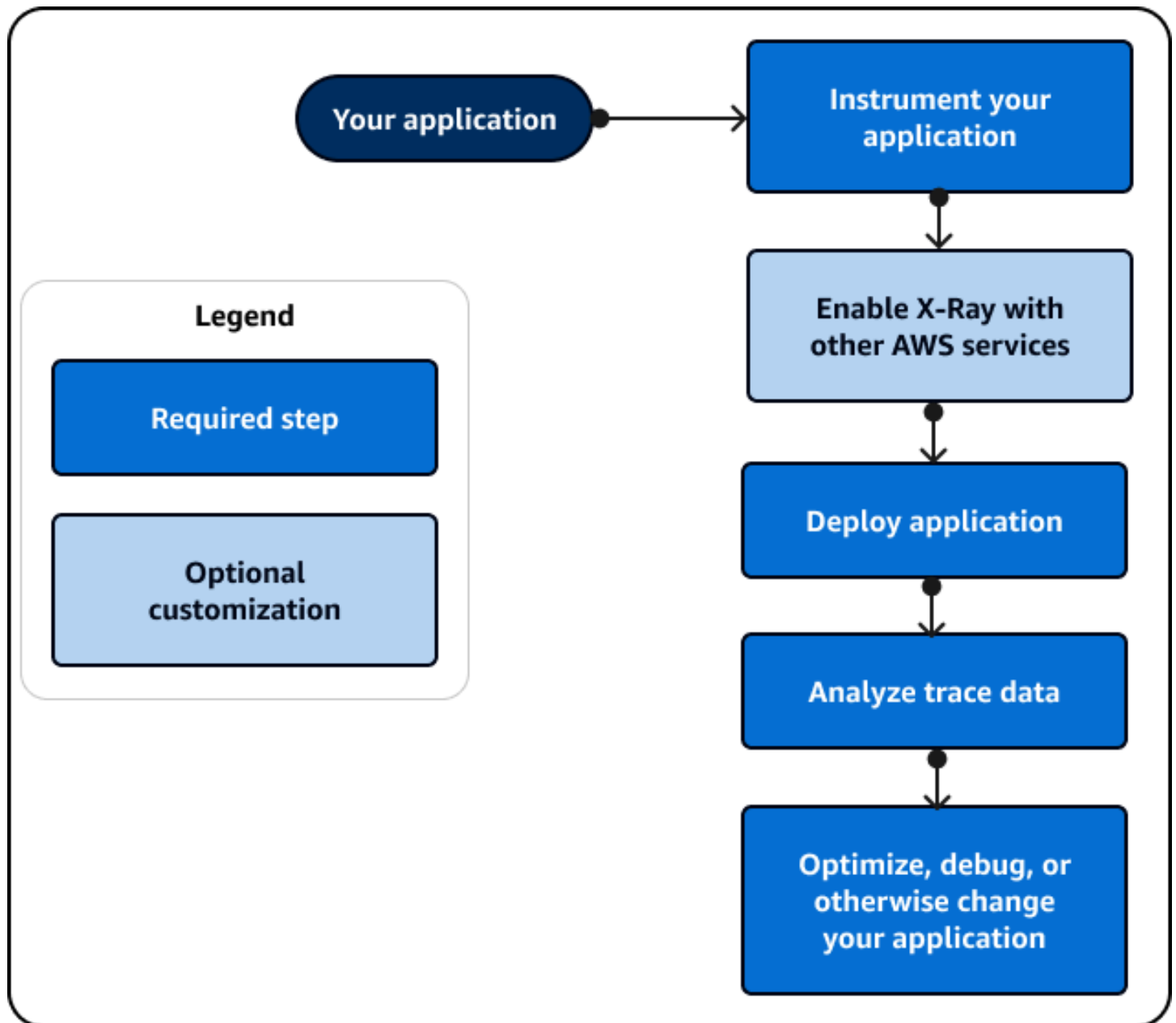
Untuk menggunakan X-Ray, lakukan langkah-langkah berikut:

1. Instrumen aplikasi Anda, yang memungkinkan X-Ray untuk melacak bagaimana aplikasi Anda memproses permintaan.
 - Gunakan X-Ray SDKs, X-Ray APIs, ADOT atau Sinyal CloudWatch Aplikasi untuk mengirim data jejak ke X-Ray. Untuk informasi selengkapnya tentang antarmuka mana yang akan digunakan, lihat [Memilih antarmuka](#).

Untuk informasi selengkapnya tentang instrumentasi, lihat [Instrumentasi aplikasi Anda untuk AWS X-Ray](#).

2. (Opsional) Konfigurasi X-Ray untuk bekerja dengan Layanan AWS yang lain yang terintegrasi dengan X-Ray. Anda dapat mengambil sampel jejak dan menambahkan header ke permintaan yang masuk, menjalankan agen atau kolektor, dan secara otomatis mengirim data jejak ke X-Ray. Untuk informasi selengkapnya, lihat [Integrasi AWS X-Ray dengan yang lain Layanan AWS](#).
3. Terapkan aplikasi instrumentasi Anda. Saat aplikasi Anda menerima permintaan, X-Ray SDK akan merekam data jejak, segmen, dan subsegmen. Pada langkah ini, Anda mungkin juga harus menyiapkan kebijakan IAM dan menyebarkan agen atau kolektor.
 - [Misalnya skrip untuk menyebarkan aplikasi menggunakan AWS Distro for OpenTelemetry \(ADOT\) SDK dan CloudWatch agen pada platform yang berbeda, lihat Skrip Demo Sinyal Aplikasi](#).
 - Untuk contoh skrip untuk menyebarkan aplikasi menggunakan X-Ray SDK dan daemon X-Ray, lihat [AWS X-Ray aplikasi sampel](#)
4. (Opsional) Buka konsol untuk melihat dan menganalisis data. Anda dapat melihat representasi GUI dari peta jejak, peta layanan, dan lainnya untuk memeriksa bagaimana fungsi aplikasi Anda. Gunakan informasi grafis di konsol untuk mengoptimalkan, men-debug, dan memahami aplikasi Anda. Untuk informasi selengkapnya tentang memilih konsol, lihat [Gunakan konsol](#).

Diagram berikut menunjukkan cara memulai menggunakan X-Ray:



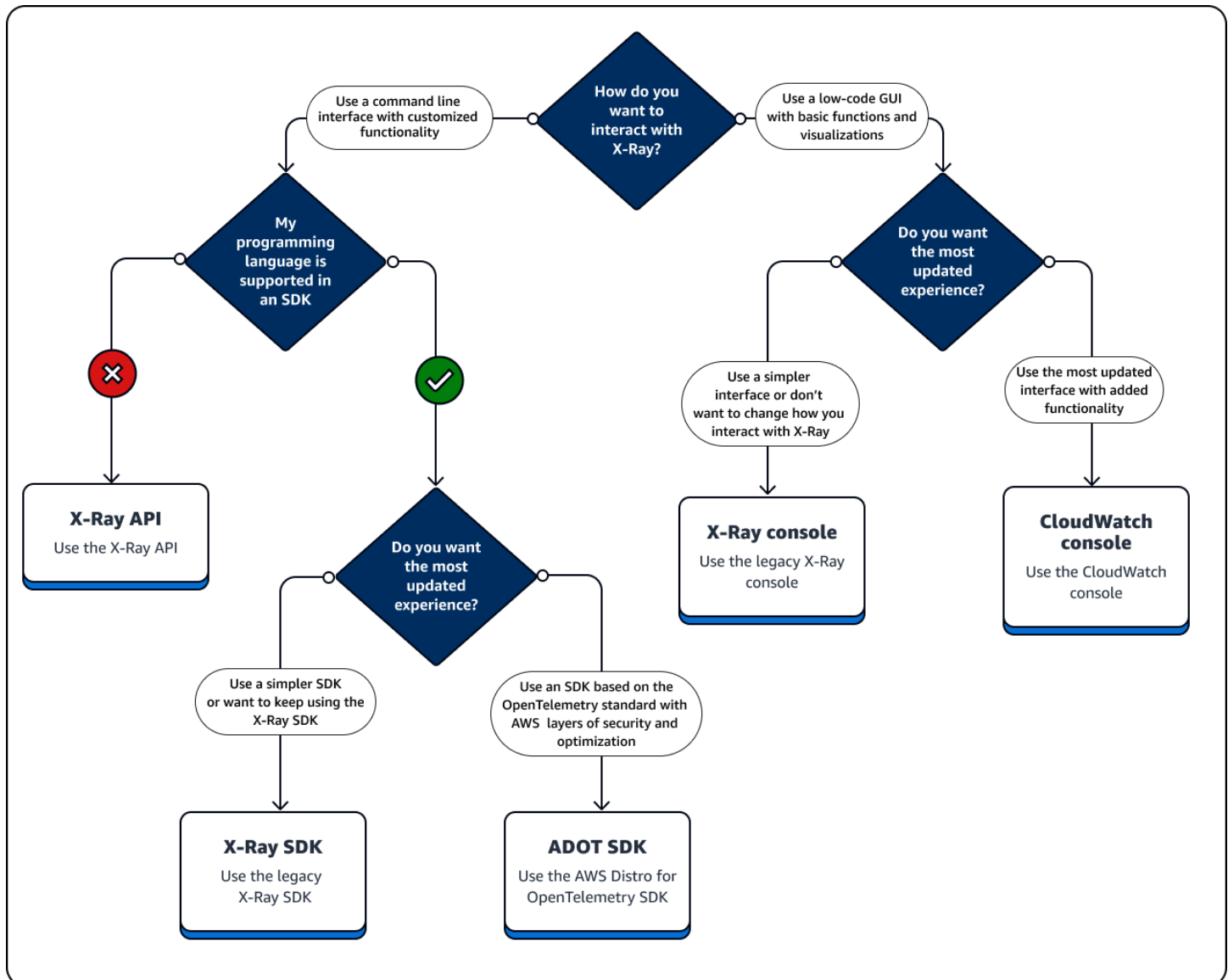
Untuk contoh data dan peta yang tersedia di konsol, luncurkan [aplikasi sampel](#) yang sudah diinstrumentasi untuk menghasilkan data jejak. Dalam beberapa menit, Anda dapat menghasilkan lalu lintas, mengirim segmen ke X-Ray, dan melihat jejak dan peta layanan.

Memilih antarmuka

AWS X-Ray dapat memberikan wawasan tentang bagaimana aplikasi Anda bekerja dan seberapa baik berinteraksi dengan layanan dan sumber daya lain. Setelah Anda menginstrumentasi atau mengonfigurasi aplikasi Anda, X-Ray mengumpulkan data jejak saat aplikasi Anda melayani permintaan. Anda dapat menganalisis data jejak ini untuk mengidentifikasi masalah kinerja, memecahkan masalah kesalahan, dan mengoptimalkan sumber daya Anda. Panduan ini menunjukkan cara berinteraksi dengan X-Ray dengan panduan berikut:

- Gunakan AWS Management Console jika Anda ingin memulai dengan cepat atau dapat menggunakan visualisasi pra-bangun untuk melakukan tugas-tugas dasar.
- Pilih CloudWatch konsol Amazon untuk pengalaman pengguna terbaru yang berisi semua fungsionalitas konsol X-Ray.
- Gunakan konsol X-Ray jika Anda menginginkan antarmuka yang lebih sederhana atau tidak ingin mengubah cara Anda berinteraksi dengan X-Ray.
- Gunakan SDK jika Anda memerlukan lebih banyak kemampuan penelusuran, pemantauan, atau pencatatan kustom daripada yang AWS Management Console dapat disediakan.
- Pilih ADOT SDK jika Anda menginginkan SDK vendor-agnostik berdasarkan open source OpenTelemetry SDK dengan lapisan AWS keamanan dan optimasi tambahan.
- Pilih X-Ray SDK jika Anda menginginkan SDK yang lebih sederhana atau tidak ingin memperbarui kode aplikasi Anda.
- Gunakan operasi X-Ray API jika SDK tidak mendukung bahasa pemrograman aplikasi Anda.

Diagram berikut membantu Anda memilih cara berinteraksi dengan X-Ray:



Jelajahi jenis antarmuka

- [Menggunakan SDK](#)
- [Gunakan konsol](#)
- [Gunakan API X-Ray](#)

Menggunakan SDK

Gunakan SDK jika Anda ingin menggunakan antarmuka baris perintah atau memerlukan lebih banyak kemampuan penelusuran, pemantauan, atau pencatatan kustom daripada yang tersedia di file. AWS Management Console Anda juga dapat menggunakan AWS SDK untuk mengembangkan

program yang menggunakan APIs X-Ray. Anda dapat menggunakan AWS Distro for OpenTelemetry (ADOT) SDK atau X-Ray SDK.

Jika Anda menggunakan SDK, Anda dapat menambahkan kustomisasi ke alur kerja Anda baik ketika Anda instrumen aplikasi Anda dan ketika Anda mengkonfigurasi kolektor atau agen Anda. Anda dapat menggunakan SDK untuk melakukan tugas-tugas berikut yang tidak dapat Anda lakukan dengan menggunakan: AWS Management Console

- Publikasikan metrik khusus — Contoh metrik pada resolusi tinggi hingga 1 detik, gunakan beberapa dimensi untuk menambahkan informasi tentang metrik, dan agregat titik data ke dalam kumpulan statistik.
- Sesuaikan kolektor Anda - Sesuaikan konfigurasi untuk setiap bagian dari kolektor termasuk penerima, prosesor, eksportir, dan konektor.
- Sesuaikan instrumentasi Anda — Sesuaikan segmen dan subsegmen, tambahkan pasangan nilai kunci khusus sebagai atribut, dan buat metrik khusus.
- Buat dan perbarui aturan pengambilan sampel secara terprogram.

Gunakan ADOT SDK jika Anda menginginkan fleksibilitas menggunakan standar OpenTelemetry SDK dengan lapisan AWS keamanan dan optimasi tambahan. AWS Distro untuk OpenTelemetry (ADOT) SDK adalah paket vendor-agnostik yang memungkinkan integrasi dengan back end dari vendor lain dan AWS non-layanan tanpa harus menyusun ulang kode Anda.

Gunakan X-Ray SDK jika Anda sudah menggunakan X-Ray SDK, hanya mengintegrasikan dengan AWS backend, dan tidak ingin mengubah cara Anda berinteraksi dengan X-Ray atau kode aplikasi Anda.

Untuk informasi selengkapnya tentang setiap fitur, lihat [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#).

Gunakan ADOT SDK

Bagian ADOT SDK adalah seperangkat open source APIs, perpustakaan, dan agen yang mengirim data ke layanan backend. ADOT didukung oleh AWS, terintegrasi dengan beberapa backend dan agen, dan menyediakan sejumlah besar pustaka sumber terbuka yang dikelola oleh OpenTelemetry komunitas. Gunakan ADOT SDK untuk menginstrumentasikan aplikasi Anda dan mengumpulkan log, metadata, metrik, dan jejak. Anda juga dapat menggunakan ADOT untuk memantau layanan dan mengatur alarm berdasarkan metrik Anda. CloudWatch

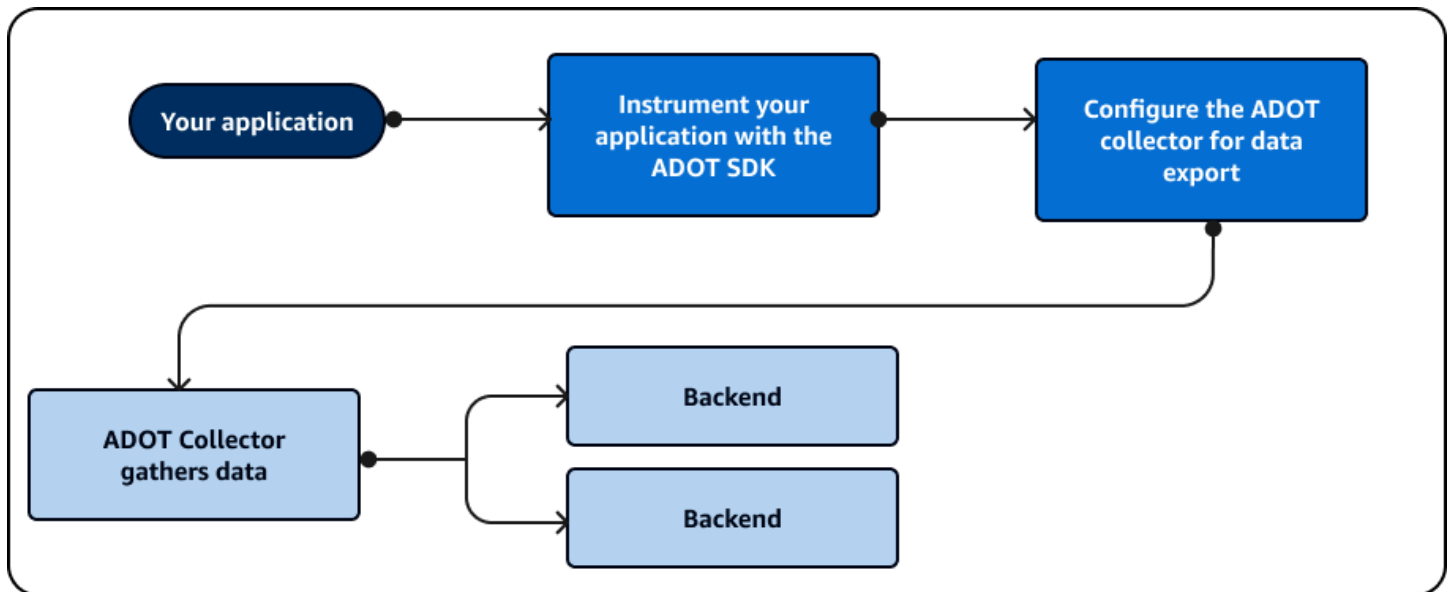
Jika Anda menggunakan ADOT SDK, Anda memiliki opsi berikut, dalam kombinasi dengan agen:

- Gunakan ADOT SDK dengan [CloudWatch agen](#) — direkomendasikan.
- Gunakan ADOT SDK dengan [ADOT Kolektor](#) — direkomendasikan jika Anda ingin menggunakan perangkat lunak agnostik vendor dengan AWS lapisan keamanan dan pengoptimalan.

Untuk menggunakan ADOT SDK, lakukan hal berikut:

- Instrumen aplikasi Anda menggunakan ADOT SDK. Untuk informasi selengkapnya, lihat dokumentasi untuk bahasa pemrograman Anda di [dokumentasi teknis ADOT](#).
- Konfigurasi sebuah ADOT kolektor untuk memberi tahu ke mana harus mengirim data yang dikumpulkannya.

Setelah ADOT kolektor menerima data Anda, ia mengirimkannya ke backend yang Anda tentukan di ADOT konfigurasi. ADOT dapat mengirim data ke beberapa backend, termasuk ke vendor di luar AWS, seperti yang ditunjukkan pada diagram berikut:



AWS update secara berkala ADOT untuk menambahkan fungsionalitas dan menyelaraskan dengan [OpenTelemetry](#) kerangka kerja. Pembaruan dan rencana masa depan untuk pengembangan ADOT adalah bagian dari [peta jalan](#) yang tersedia untuk umum. ADOT mendukung beberapa bahasa pemrograman yang meliputi:

- Go
- Java

- JavaScript
- Python
- .NET
- Ruby
- PHP

Jika Anda menggunakan Python, ADOT dapat secara otomatis instrumen aplikasi Anda. Untuk mulai menggunakan ADOT, lihat [Pendahuluan](#) dan [Memulai dengan AWS Distro untuk OpenTelemetry Kolektor](#).

Gunakan X-Ray SDK

X-Ray SDK adalah sekumpulan AWS APIs dan pustaka yang mengirim data ke layanan AWS backend. Gunakan X-Ray SDK untuk instrumen aplikasi Anda dan mengumpulkan data jejak. Anda tidak dapat menggunakan X-Ray SDK untuk mengumpulkan data log atau metrik.

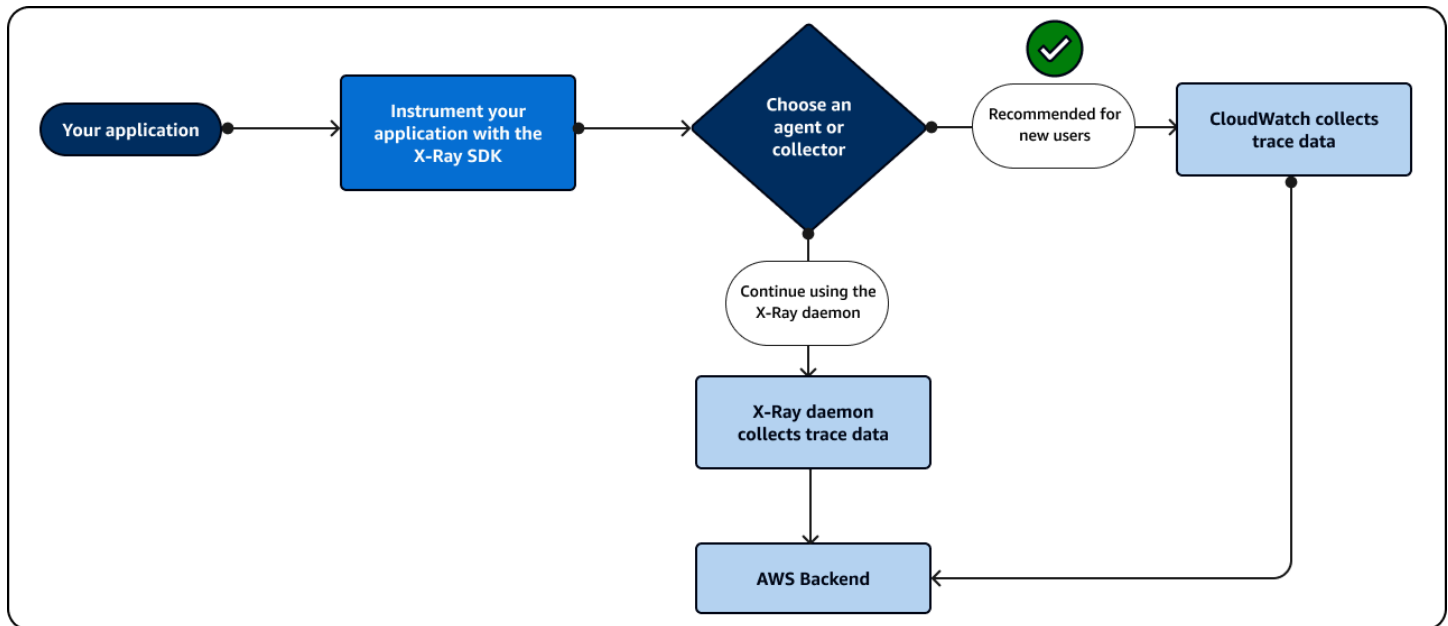
Jika Anda menggunakan X-Ray SDK, Anda memiliki opsi berikut, dalam kombinasi dengan agen:

- Gunakan X-Ray SDK dengan [AWS X-Ray daemon](#) — Gunakan ini jika Anda tidak ingin memperbarui kode aplikasi Anda.
- Gunakan X-Ray SDK dengan CloudWatch agen — (disarankan) CloudWatch Agen ini kompatibel dengan X-Ray SDK.

Untuk menggunakan X-Ray SDK, lakukan hal berikut:

- Instrumen aplikasi Anda menggunakan X-Ray SDK.
- Konfigurasi kolektor untuk memberi tahu ke mana harus mengirim data yang dikumpulkannya. Anda dapat menggunakan CloudWatch agen atau daemon X-Ray untuk mengumpulkan informasi jejak Anda.

Setelah kolektor atau agen menerima data Anda, ia mengirimkannya ke AWS backend yang Anda tentukan dalam konfigurasi agen. X-Ray SDK hanya dapat mengirim data ke AWS backend seperti yang ditunjukkan pada diagram berikut:



Jika Anda menggunakan Java, Anda dapat menggunakan X-Ray SDK untuk secara otomatis instrumen aplikasi Anda. Untuk mulai menggunakan X-Ray SDK, lihat pustaka yang terkait dengan bahasa pemrograman berikut:

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

Gunakan konsol

Gunakan konsol jika Anda menginginkan antarmuka pengguna grafis (GUI) yang membutuhkan pengkodean minimal. Pengguna yang baru mengenal X-Ray dapat memulai dengan cepat menggunakan visualisasi pra-bangun, dan melakukan tugas-tugas dasar. Anda dapat melakukan hal berikut langsung dari konsol:

- Aktifkan X-Ray.
- Lihat ringkasan performa aplikasi tingkat tinggi.
- Periksa status kesehatan aplikasi Anda.

- Identifikasi kesalahan tingkat tinggi.
- Lihat ringkasan jejak dasar.

Anda dapat menggunakan CloudWatch konsol Amazon di <https://console.aws.amazon.com/cloudwatch/> atau konsol X-Ray di <https://console.aws.amazon.com/xray/rumah> untuk berinteraksi dengan X-Ray.

Gunakan CloudWatch konsol Amazon

CloudWatch Konsol ini mencakup fungsionalitas X-Ray baru yang didesain ulang dari konsol X-Ray agar lebih mudah digunakan. Jika Anda menggunakan CloudWatch konsol, Anda dapat melihat CloudWatch log dan metrik bersama dengan data jejak X-Ray. Gunakan CloudWatch konsol untuk melihat dan menganalisis data termasuk yang berikut:

- Jejak X-Ray - Lihat, analisis, dan filter jejak yang terkait dengan aplikasi Anda saat melayani permintaan. Gunakan jejak ini untuk menemukan latensi tinggi, kesalahan debug, dan mengoptimalkan alur kerja aplikasi Anda. Lihat peta jejak dan peta layanan untuk melihat representasi visual alur kerja aplikasi Anda.
- Log — Lihat, menganalisis, dan memfilter log yang dihasilkan aplikasi Anda. Gunakan log untuk memecahkan masalah kesalahan dan mengatur pemantauan berdasarkan nilai log tertentu.
- Metrik — Ukur dan pantau performa aplikasi Anda menggunakan metrik yang dipancarkan sumber daya Anda atau buat metrik Anda sendiri. Lihat metrik ini dalam grafik dan bagan.
- Memantau jaringan dan infrastruktur - Memantau jaringan utama untuk pemadaman dan kesehatan dan kinerja infrastruktur Anda termasuk aplikasi dalam peti kemas, AWS layanan lain, dan klien.
- Semua fungsionalitas dari konsol X-Ray yang tercantum dalam berikut Gunakan bagian konsol X-Ray.

Untuk informasi selengkapnya tentang CloudWatch konsol, lihat [Memulai Amazon CloudWatch](#).

Masuk ke CloudWatch konsol Amazon di <https://console.aws.amazon.com/cloudwatch/>.

Gunakan konsol X-Ray

Konsol X-Ray menawarkan penelusuran terdistribusi untuk permintaan aplikasi. Gunakan konsol X-Ray jika Anda menginginkan pengalaman konsol yang lebih sederhana atau tidak ingin memperbarui kode aplikasi Anda. AWS tidak lagi mengembangkan konsol X-Ray. Konsol X-Ray berisi fitur-fitur berikut untuk aplikasi berinstrumen:

- [Wawasan](#) — Secara otomatis mendeteksi anomali dalam kinerja aplikasi Anda dan menemukan penyebab yang mendasarinya. Wawasan disertakan dalam CloudWatch konsol di bawah Wawasan. Untuk informasi selengkapnya, lihat Use X-Ray Insights di [Gunakan konsol X-Ray](#).
- Peta layanan — Lihat struktur grafis aplikasi Anda dan hubungannya dengan klien, sumber daya, layanan, dan dependensi.
- Jejak — Lihat ikhtisar jejak yang dihasilkan oleh aplikasi Anda saat melayani permintaan. Gunakan data jejak untuk memahami kinerja aplikasi Anda terhadap metrik dasar termasuk HTTP respon dan waktu respon.
- Analytics — Menafsirkan, mengeksplorasi, dan menganalisis data jejak menggunakan grafik untuk distribusi waktu respons.
- Konfigurasi - Buat jejak yang disesuaikan untuk mengubah konfigurasi default sebagai berikut:
 - Sampling — Buat aturan yang menentukan seberapa sering mengambil sampel aplikasi Anda untuk informasi pelacakan. Untuk informasi selengkapnya, lihat Mengonfigurasi aturan pengambilan sampel di [Gunakan konsol X-Ray](#).
 - [Enkripsi](#) — Enkripsi data saat istirahat menggunakan kunci yang dapat Anda audit atau nonaktifkan menggunakan AWS Key Management Service.
 - Grup — Gunakan ekspresi filter untuk menentukan sekelompok jejak dengan fitur umum seperti nama url atau waktu respons. Untuk informasi selengkapnya, lihat [Mengkonfigurasi grup](#).

Login konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.

Jelajahi konsol X-Ray

Gunakan konsol X-Ray untuk melihat peta layanan dan jejak terkait untuk permintaan yang disajikan aplikasi Anda, dan untuk mengonfigurasi grup dan aturan pengambilan sampel yang memengaruhi cara jejak dikirim ke X-Ray.

Note

Peta dan CloudWatch ServiceLens peta Layanan X-Ray telah digabungkan ke dalam peta jejak X-Ray di dalam CloudWatch konsol Amazon. Buka [CloudWatchkonsol](#) dan pilih Trace Map di bawah jejak X-Ray dari panel navigasi kiri.

CloudWatch sekarang termasuk [Application Signals](#), yang dapat menemukan dan memantau layanan aplikasi Anda, klien, kenari Synthetics, dan dependensi layanan. Gunakan Sinyal Aplikasi untuk melihat daftar atau peta visual layanan Anda, melihat metrik kesehatan

berdasarkan sasaran tingkat layanan Anda (SLOs), dan menelusuri untuk melihat jejak X-Ray yang berkorelasi untuk pemecahan masalah yang lebih rinci.

Halaman konsol X-Ray utama adalah peta jejak, yang merupakan representasi visual dari grafik layanan JSON yang dihasilkan X-Ray dari data jejak yang dihasilkan oleh aplikasi Anda. Peta terdiri dari simpul layanan untuk setiap aplikasi di akun Anda yang melayani permintaan, simpul klien hulu yang mewakili asal-usul permintaan, dan simpul layanan hilir yang mewakili layanan web dan sumber daya yang digunakan oleh aplikasi saat memproses permintaan. Ada halaman tambahan untuk melihat jejak dan detail jejak, dan mengonfigurasi grup dan aturan pengambilan sampel.

Lihat pengalaman konsol untuk X-Ray dan bandingkan dengan CloudWatch konsol di bagian berikut.

Jelajahi X-Ray dan CloudWatch konsol

- [Menggunakan peta jejak X-Ray](#)
- [Melihat jejak dan detail jejak](#)
- [Menggunakan ekspresi filter](#)
- [Penelusuran lintas akun](#)
- [Menelusuri aplikasi berbasis peristiwa](#)
- [Menggunakan histogram latensi](#)
- [Menggunakan wawasan X-Ray](#)
- [Berinteraksi dengan konsol Analitik](#)
- [Mengkonfigurasi grup](#)
- [Mengonfigurasi aturan pengambilan sampel](#)
- [Konsol deep linking](#)

Menggunakan peta jejak X-Ray

Lihat peta jejak X-Ray untuk mengidentifikasi layanan di mana terjadi kesalahan, koneksi dengan latensi tinggi, atau jejak permintaan yang tidak berhasil.

Note

CloudWatch sekarang termasuk [Sinyal Aplikasi](#), yang dapat menemukan dan memantau layanan aplikasi Anda, klien, kenari sintetis, dan dependensi layanan. Gunakan Sinyal

Aplikasi untuk melihat daftar atau peta visual layanan Anda, melihat metrik kesehatan berdasarkan sasaran tingkat layanan Anda (SLOs), dan menelusuri untuk melihat jejak X-Ray yang berkorelasi untuk pemecahan masalah yang lebih rinci.

Peta dan CloudWatch ServiceLens peta layanan X-Ray digabungkan ke dalam peta jejak X-Ray di dalam CloudWatch konsol Amazon. Buka [CloudWatch konsol](#) dan pilih Trace Map di bawah jejak X-Ray dari panel navigasi kiri.

Melihat peta pelacakan

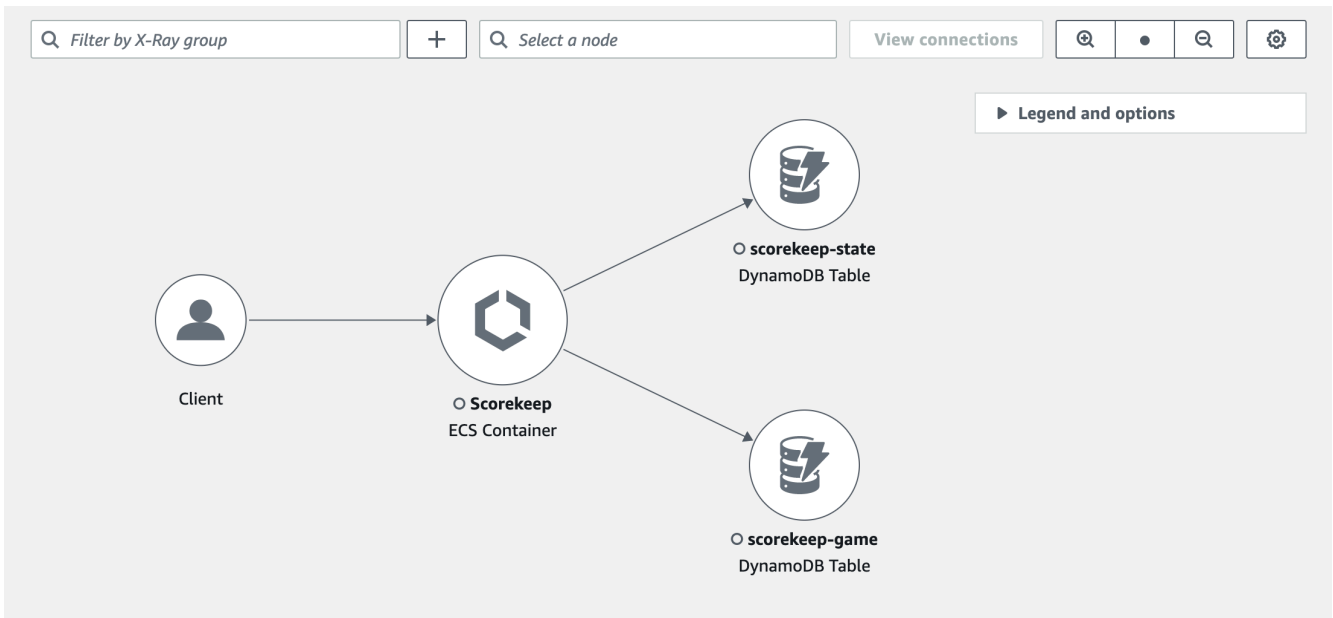
Peta jejak adalah representasi visual dari data jejak yang dihasilkan oleh aplikasi Anda. Peta menunjukkan node layanan yang melayani permintaan, node klien hulu yang mewakili asal permintaan, dan node layanan hilir yang mewakili layanan web dan sumber daya yang digunakan oleh aplikasi saat memproses permintaan.

Peta jejak menampilkan tampilan jejak yang terhubung di seluruh aplikasi berbasis peristiwa yang menggunakan Amazon SQS dan Lambda. Untuk informasi selengkapnya, lihat [melacak aplikasi berbasis peristiwa](#). Peta jejak juga mendukung [penelusuran lintas akun](#), menampilkan node dari beberapa akun dalam satu peta.

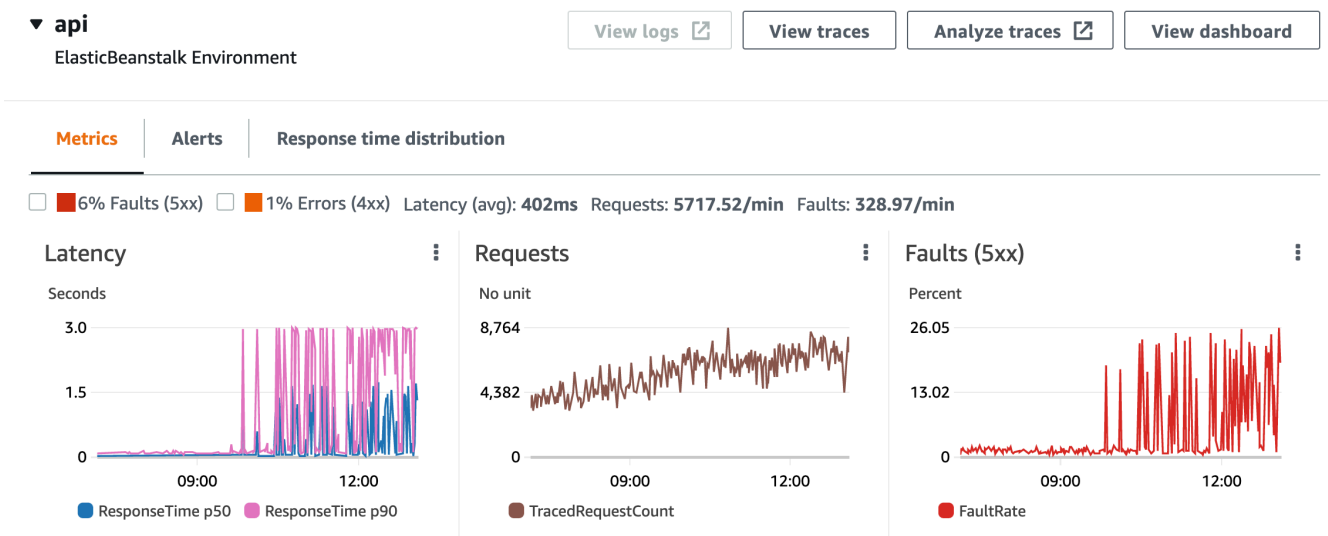
CloudWatch console

Untuk melihat peta jejak di CloudWatch konsol

1. Buka [konsol CloudWatch](#) . Pilih Trace Map di bawah bagian X-Ray Traces di panel navigasi kiri.



- Pilih simpul layanan untuk melihat permintaan untuk simpul tersebut, atau edge antara dua simpul untuk melihat permintaan yang melintasi koneksi tersebut.
- Informasi tambahan ditampilkan di bawah peta jejak, termasuk tab untuk metrik, peringatan, dan distribusi waktu respons. Pada tab Metrik, pilih rentang dalam setiap grafik untuk menelusuri untuk melihat lebih detail, atau pilih opsi Kesalahan atau Kesalahan untuk memfilter jejak. Pada tab Distribusi waktu respons, pilih rentang dalam grafik untuk memfilter jejak berdasarkan waktu respons.



- Lihat jejak dengan memilih Lihat jejak, atau jika filter telah diterapkan, pilih Lihat jejak yang difilter.

5. Pilih Lihat log untuk melihat CloudWatch log yang terkait dengan node yang dipilih. Tidak semua node peta jejak mendukung log tampilan. Lihat [CloudWatch log pemecahan masalah](#) untuk informasi selengkapnya.

Peta jejak menunjukkan masalah dalam setiap node dengan menguraikannya dengan warna:

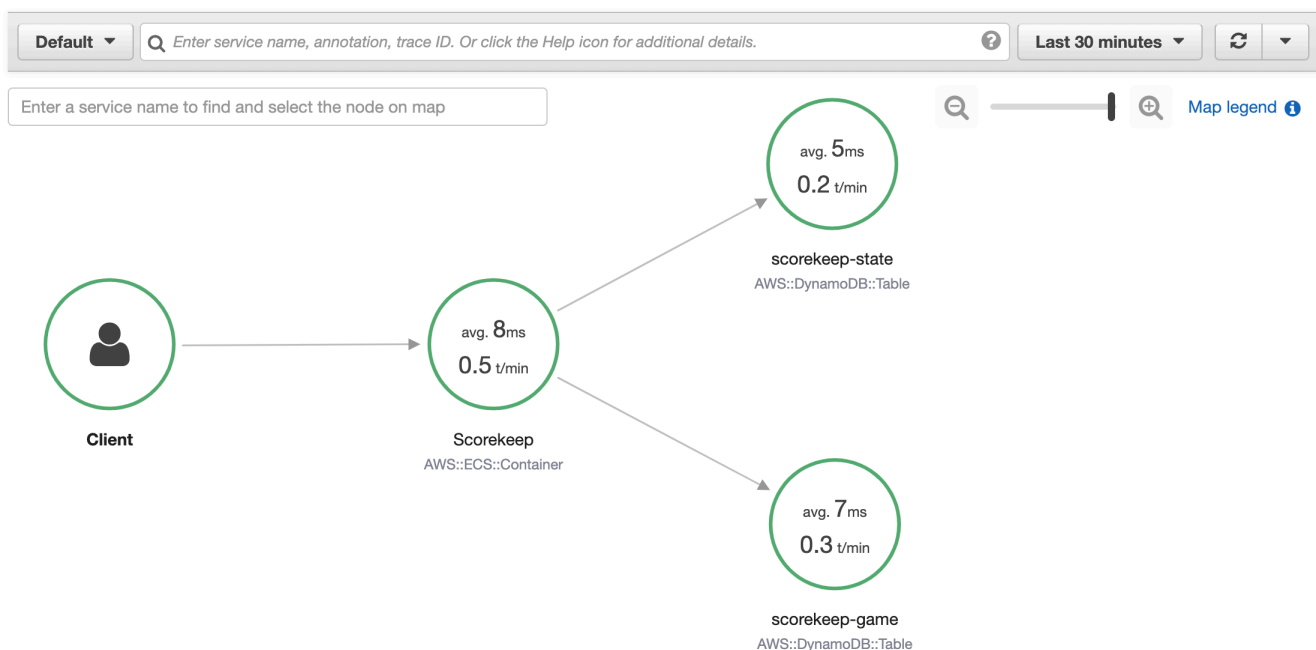
- Merah untuk kesalahan server (500 kesalahan seri)
- Kuning untuk kesalahan klien (400 kesalahan seri)
- Ungu untuk kesalahan throttling (429 Terlalu Banyak Permintaan)

Jika peta jejak Anda besar, gunakan kontrol atau mouse di layar untuk memperbesar dan memperkecil dan memindahkan peta.

X-Ray console

Untuk melihat peta Layanan

1. Buka [konsol X-Ray](#). Peta layanan ditampilkan secara default. Anda juga dapat memilih Peta Layanan dari panel navigasi kiri.



2. Pilih simpul layanan untuk melihat permintaan untuk simpul tersebut, atau edge antara dua simpul untuk melihat permintaan yang melintasi koneksi tersebut.

3. Gunakan [histogram](#) distribusi respons untuk memfilter jejak berdasarkan durasi, dan pilih kode status yang ingin Anda lihat jejaknya. Kemudian pilih Melihat pelacakan untuk membuka daftar pelacakan dengan menerapkan ekspresi filter.

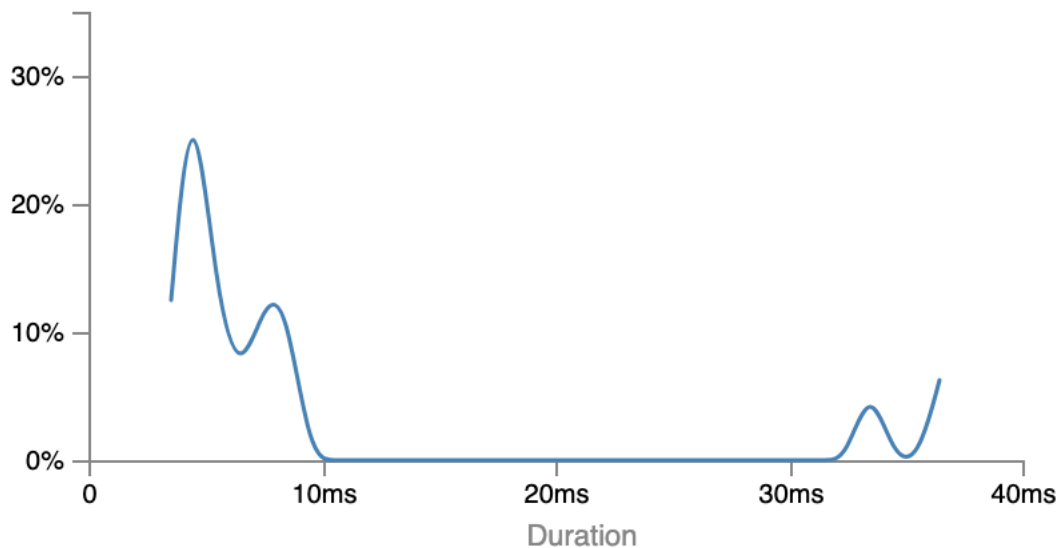
Service details ?

Name: Scorekeep

Type: AWS::ECS::Container

Response distribution

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



Response status

Choose response statuses to add to the filter when viewing traces.

☐ Fault: 0%

☐ Error: 0%

☐ Throttle: 0%

☐ OK: 100%

Analyze traces

View traces >

Peta layanan menunjukkan kondisi setiap simpul dengan mewarnainya berdasarkan rasio panggilan sukses ke kesalahan dan kesalahan:

- Hijau untuk panggilan berhasil
- Merah untuk kesalahan server (500 kesalahan seri)
- Kuning untuk kesalahan klien (400 kesalahan seri)
- Ungu untuk kesalahan throttling (429 Terlalu Banyak Permintaan)

Jika peta layanan Anda besar, gunakan kontrol atau mouse di layar untuk memperbesar dan memperkecil dan memindahkan peta.

Note

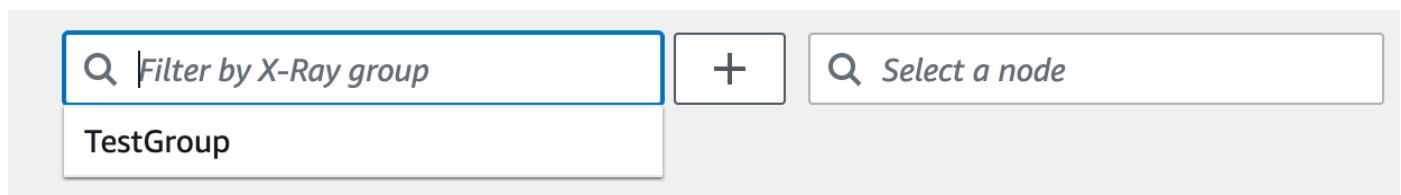
Peta jejak X-Ray dapat menampilkan hingga 10.000 node. Dalam skenario langka di mana jumlah total node layanan melebihi batas ini, Anda mungkin menerima kesalahan dan tidak dapat menampilkan peta jejak lengkap di konsol.

Memfilter peta jejak menurut grup

Menggunakan [ekspresi filter](#), Anda dapat menentukan kriteria yang digunakan untuk menyertakan jejak dalam grup. Gunakan langkah-langkah berikut untuk kemudian menampilkan grup tertentu di peta jejak.

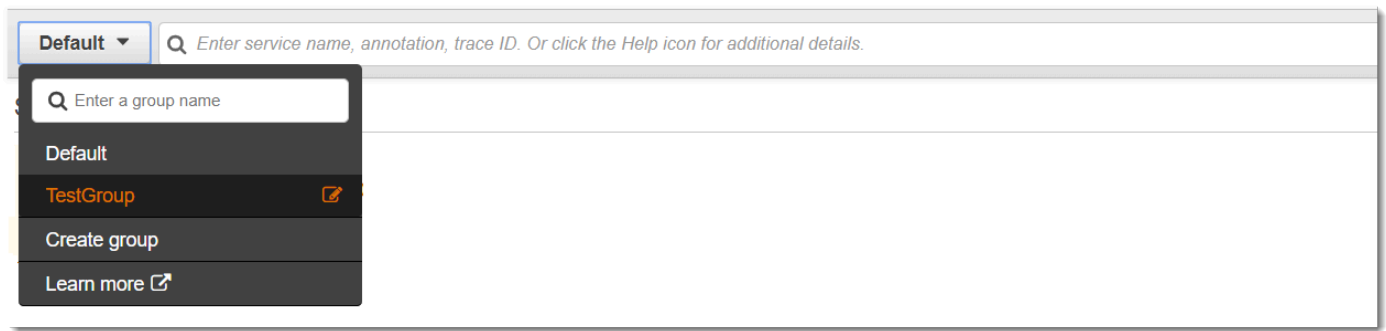
CloudWatch console

Pilih nama grup dari filter grup di kiri atas peta jejak.



X-Ray console

Pilih nama grup dari menu tarik-turun di sebelah kiri bilah pencarian.



Peta layanan sekarang akan difilter untuk menampilkan jejak yang cocok dengan ekspresi filter dari grup yang dipilih.

Lacak legenda dan opsi peta

Peta jejak mencakup legenda dan beberapa opsi untuk menyesuaikan tampilan peta.

CloudWatch console

Pilih Legenda dan opsi drop-down di kanan atas peta. Pilih apa yang ditampilkan dalam node, termasuk:

- Metrik menampilkan waktu respons rata-rata dan jumlah jejak yang dikirim per menit selama rentang waktu yang dipilih.
- Node menampilkan ikon layanan dalam setiap node.

Pilih pengaturan peta tambahan dari panel Preferensi, yang dapat diakses melalui ikon roda gigi di kanan atas peta. Pengaturan ini termasuk memilih metrik mana yang digunakan untuk menentukan ukuran setiap node, dan kenari mana yang harus ditampilkan di peta.

X-Ray console

Tampilkan legenda peta layanan dengan memilih tautan Legenda peta di kanan atas peta. Opsi peta layanan dapat dipilih di kanan bawah peta jejak, termasuk:

- Ikon Layanan mengubah apa yang ditampilkan dalam setiap node, menampilkan ikon layanan, atau waktu respons rata-rata dan jumlah jejak yang dikirim per menit selama rentang waktu yang dipilih.
- Ukuran node: Tidak ada yang mengatur semua node ke ukuran yang sama.

- Ukuran node: Health mengukur node berdasarkan jumlah permintaan yang terkena dampak termasuk kesalahan, kesalahan, atau permintaan yang dibatasi.
- Ukuran node: Ukuran lalu lintas node dengan jumlah total permintaan.

Melihat jejak dan detail jejak

Gunakan halaman Jejak di konsol X-Ray untuk menemukan jejak berdasarkan URL, kode respons, atau data lain dari ringkasan jejak. Setelah memilih jejak dari daftar jejak, halaman Detail jejak menampilkan peta node layanan yang terkait dengan jejak yang dipilih dan garis waktu segmen jejak.

Melihat pelacakan

CloudWatch console

Untuk melihat jejak di CloudWatch konsol

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Di panel navigasi kiri, pilih jejak X-Ray, lalu pilih Jejak. Anda dapat memfilter berdasarkan grup atau memasukkan [ekspresi filter](#). Ini menyaring jejak yang ditampilkan di bagian Jejak di bagian bawah halaman.

Atau, Anda dapat menggunakan peta layanan untuk menavigasi ke node layanan tertentu, dan kemudian melihat jejak. Ini membuka halaman Jejak dengan kueri yang sudah diterapkan.

3. Perbaiki kueri Anda di bagian Query refiners. Untuk memfilter jejak berdasarkan atribut umum, pilih opsi dari panah bawah di sebelah Refine query by. Pilihannya meliputi yang berikut:
 - Node - Filter jejak berdasarkan node layanan.
 - ARN Sumber Daya — Filter jejak berdasarkan sumber daya yang terkait dengan jejak. Contoh sumber daya ini termasuk instans Amazon Elastic Compute Cloud (Amazon EC2), AWS Lambda fungsi, atau Amazon DynamoDB tabel.
 - Pengguna - Filter jejak dengan ID pengguna.
 - Pesan akar penyebab kesalahan - Filter jejak berdasarkan akar penyebab kesalahan.
 - URL — Filter jejak berdasarkan jalur URL yang digunakan oleh aplikasi Anda.

- Kode status HTTP - Filter jejak dengan kode status HTTP yang dikembalikan oleh aplikasi Anda. Anda dapat menentukan kode respons khusus atau memilih dari berikut ini:
 - 200—Permintaan itu berhasil.
 - 401— Permintaan tidak memiliki kredensi otentikasi yang valid.
 - 403— Permintaan tidak memiliki izin yang valid.
 - 404— Server tidak dapat menemukan sumber daya yang diminta.
 - 500— Server mengalami kondisi yang tidak terduga dan menghasilkan kesalahan internal.

Pilih satu atau beberapa entri lalu pilih Tambahkan ke kueri untuk ditambahkan ke ekspresi filter di bagian atas halaman.

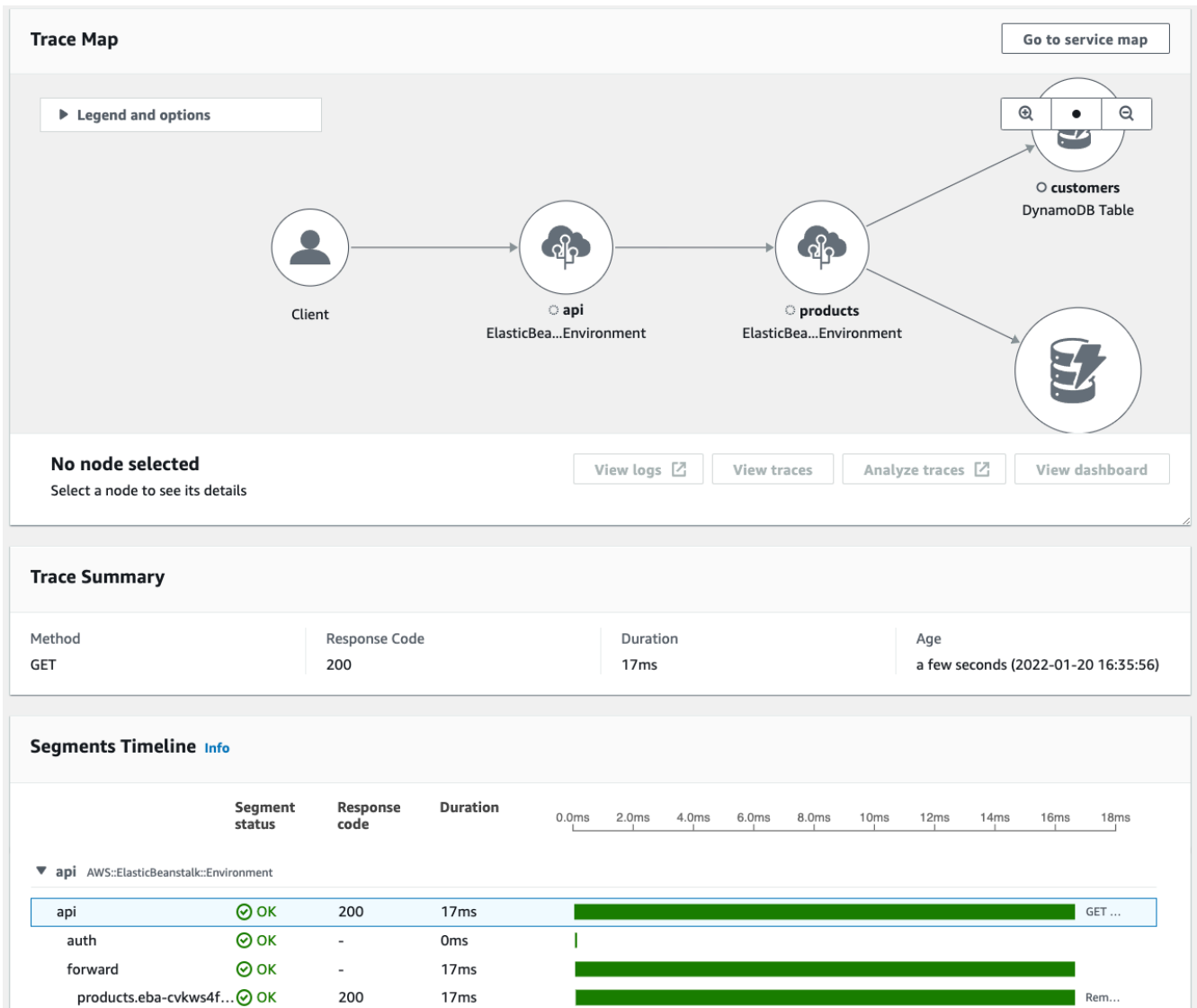
4. Untuk menemukan jejak tunggal, masukkan [ID jejak](#) langsung ke bidang kueri. Anda dapat menggunakan format X-Ray atau format World Wide Web Consortium (W3C). Misalnya, jejak yang dibuat menggunakan [AWS Distro for OpenTelemetry](#) dalam format W3C.

Note

Saat Anda menanyakan jejak yang dibuat dengan ID jejak format W3C, konsol menampilkan jejak yang cocok dalam format X-Ray. Misalnya, jika Anda melakukan kueri 4efaaf4d1e8720b39541901950019ee5 dalam format W3C, konsol akan menampilkan setara X-Ray: 1-4efaaf4d-1e8720b39541901950019ee5

5. Pilih Jalankan kueri kapan saja untuk menampilkan daftar jejak yang cocok dalam bagian Jejak di bagian bawah halaman.
6. Untuk menampilkan halaman Detail jejak untuk satu jejak, pilih ID jejak dari daftar.

Gambar berikut menunjukkan peta Trace yang berisi node layanan yang terkait dengan jejak dan tepi antara node yang mewakili jalur yang diambil oleh segmen yang menyusun jejak. Ringkasan Jejak mengikuti Peta Jejak. Ringkasan berisi informasi tentang GET operasi sampel, Kode Responsnya, Durasi yang diperlukan jejak untuk dijalankan, dan Usia permintaan. Garis Waktu Segmen mengikuti Ringkasan Jejak yang menunjukkan durasi segmen dan subsegmen jejak.



Jika Anda memiliki aplikasi berbasis peristiwa yang menggunakan Amazon SQS dan Lambda, Anda dapat melihat tampilan jejak yang terhubung untuk setiap permintaan di peta Trace. Dalam peta, jejak dari produsen pesan terkait dengan jejak dari AWS Lambda konsumen dan ditampilkan sebagai tepi garis putus-putus. Untuk informasi selengkapnya tentang aplikasi berbasis peristiwa, lihat. [Menelusuri aplikasi berbasis peristiwa](#)

Halaman detail Traces and Trace juga mendukung [penelusuran lintas akun](#), yang dapat mencantumkan jejak dari beberapa akun dalam daftar jejak dan di dalam satu peta jejak.

X-Ray console

Untuk melihat jejak di konsol X-Ray

1. Buka halaman [Jejak](#) di konsol X-Ray. Panel ikhtisar Trace menampilkan daftar jejak yang dikelompokkan berdasarkan fitur umum termasuk Akar penyebab kesalahan, ResourceArn, dan InstanceId
2. Untuk memilih fitur umum untuk melihat kumpulan jejak yang dikelompokkan, perluas panah bawah di samping Kelompokkan menurut. Ilustrasi berikut menunjukkan gambaran jejak jejak yang dikelompokkan berdasarkan URL untuk [AWS X-Ray aplikasi sampel](#), dan daftar jejak terkait.

Trace overview

Group by: URL

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

Trace list (21)

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

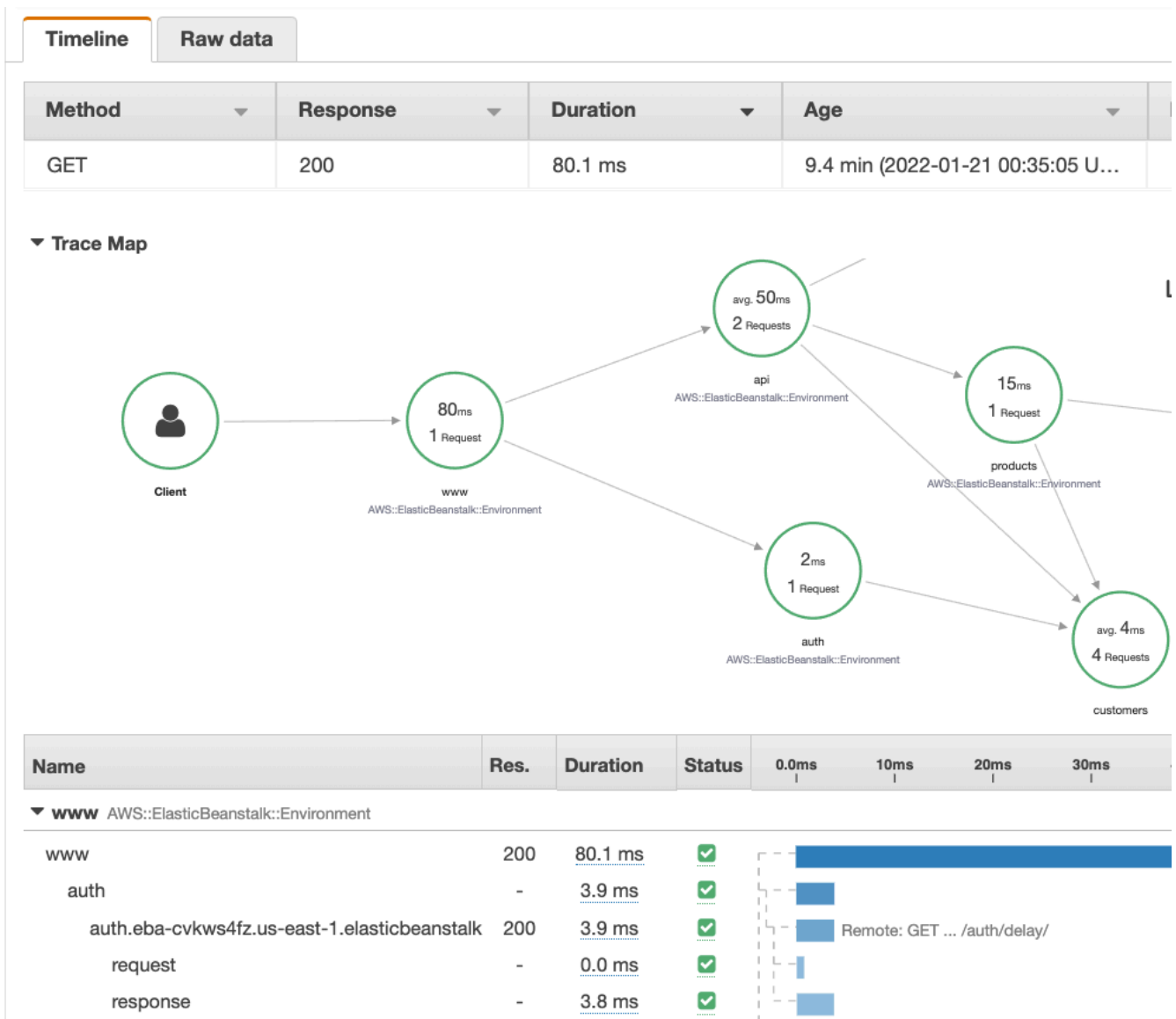
3. Pilih ID jejak untuk melihatnya di bawah daftar Trace. Anda juga dapat memilih Peta layanan di panel navigasi untuk melihat jejak untuk node layanan tertentu. Kemudian Anda dapat melihat jejak yang terkait dengan node itu.

Tab Timeline menunjukkan alur permintaan untuk pelacakan, dan menyertakan yang berikut:

- Peta jalur untuk setiap segmen dalam jejak.

- Berapa lama waktu yang dibutuhkan segmen untuk mencapai node di peta jejak.
- Berapa banyak permintaan yang dibuat ke node di peta jejak.

Ilustrasi berikut menunjukkan contoh Trace Map terkait dengan GET permintaan yang dibuat untuk aplikasi sampel. Panah menunjukkan jalur yang diambil setiap segmen untuk menyelesaikan permintaan. Node layanan menunjukkan jumlah permintaan yang dibuat selama GET permintaan.



Untuk informasi selengkapnya tentang tab Timeline, lihat bagian Menjelajahi garis waktu jejak berikut.

Tab data mentah menunjukkan informasi tentang jejak, dan segmen serta subsegmen yang menyusun jejak, dalam JSON format. Informasi ini mungkin termasuk yang berikut:

- Stempel waktu
- Unik IDs
- Sumber daya yang terkait dengan segmen atau subsegmen
- Sumber, atau asal, segmen atau subsegmen
- Informasi tambahan tentang permintaan ke aplikasi Anda seperti respons dari permintaan HTTP

Menjelajahi garis waktu jejak

Bagian Timeline menunjukkan hierarki segmen dan subsegmen di sebelah bilah horizontal yang sesuai dengan waktu yang mereka gunakan untuk menyelesaikan tugas mereka. Entri pertama dalam daftar adalah segmen, yang mewakili semua data yang dicatat oleh layanan untuk satu permintaan. Subsegmen diindentasi dan terdaftar mengikuti segmen. Kolom berisi informasi tentang setiap segmen.

CloudWatch console

Di CloudWatch konsol, Timeline Segments menyediakan informasi berikut:

- Kolom pertama: Daftar segmen dan subsegmen dalam jejak yang dipilih.
- Kolom status Segmen: Daftar hasil status dari setiap segmen dan subsegmen.
- Kolom kode Respons: Daftar kode status respons HTTP ke permintaan browser yang dibuat oleh segmen atau subsegmen, bila tersedia.
- Kolom Durasi: Daftar berapa lama segmen atau subsegmen berjalan.
- Kolom Hosted in: Daftar namespace atau lingkungan tempat segmen atau subsegmen dijalankan, jika berlaku. Untuk informasi selengkapnya, lihat [Dimensi yang dikumpulkan dan kombinasi dimensi](#).
- Kolom terakhir: Menampilkan bilah horizontal yang sesuai dengan durasi segmen atau subsegmen dijalankan, sehubungan dengan segmen atau subsegmen lain dalam garis waktu.

Untuk mengelompokkan daftar segmen dan subsegmen berdasarkan node layanan, aktifkan Grup menurut node.

X-Ray console

Di halaman detail jejak, pilih tab Timeline untuk melihat garis waktu untuk setiap segmen dan subsegmen yang membentuk jejak.

Di konsol X-Ray, Timeline memberikan informasi berikut:

- Kolom Nama: Daftar nama segmen dan subsegmen dalam jejak.
- Kolom Res.: Daftar kode status respons HTTP ke permintaan browser yang dibuat oleh segmen atau subsegmen, bila tersedia.
- Kolom Durasi: Daftar berapa lama segmen atau subsegmen berjalan.
- Kolom Status: Daftar hasil dari status segmen atau subsegmen.
- Kolom terakhir: Menampilkan bilah horizontal yang sesuai dengan durasi segmen atau subsegmen dijalankan, sehubungan dengan segmen atau subsegmen lain dalam garis waktu.

Untuk melihat data jejak mentah yang digunakan konsol untuk menghasilkan timeline, pilih tab Data mentah. Data mentah menunjukkan informasi tentang jejak, serta segmen serta subsegmen yang menyusun jejak dalam JSON format. Informasi ini mungkin termasuk yang berikut:

- Stempel waktu
- Unik IDs
- Sumber daya yang terkait dengan segmen atau subsegmen
- Sumber, atau asal, segmen atau subsegmen
- Informasi tambahan tentang permintaan ke aplikasi Anda seperti respons dari permintaan HTTP.

Saat Anda menggunakan AWS SDK yang diinstrumentasi, HTTP, atau SQL klien untuk melakukan panggilan ke sumber daya eksternal, X-Ray SDK merekam subsegmen secara otomatis. Anda juga dapat menggunakan X-Ray SDK untuk merekam subsegmen kustom untuk fungsi atau blok kode apa pun. Subsegmen tambahan yang direkam saat subsegmen kustom terbuka menjadi turunan dari subsegmen kustom.

Melihat detail segmen

Dari garis waktu jejak, pilih nama segmen untuk melihat detailnya.

Panel detail Segmen menampilkan tab Ikhtisar, Sumber Daya, Anotasi, Metadata, Pengecualian, dan SQL. Berikut ini berlaku:

- Tab Gambaran Umum menampilkan informasi tentang permintaan dan respons. Informasi mencakup nama, waktu mulai, waktu akhir, durasi, URL permintaan, operasi permintaan, kode respons permintaan, dan kesalahan dan kesalahan apa pun.
- Tab Sumber Daya untuk segmen menampilkan informasi dari X-Ray SDK dan tentang AWS sumber daya yang menjalankan aplikasi Anda. Gunakan plugin Amazon EC2 AWS Elastic Beanstalk, atau Amazon ECS untuk X-Ray SDK untuk merekam informasi sumber daya khusus layanan. Untuk informasi selengkapnya tentang plugin, lihat bagian Plugin layanan di [Mengonfigurasi X-Ray SDK for Java](#)
- Tab yang tersisa menampilkan Anotasi, Metadata, dan Pengecualian yang direkam untuk segmen tersebut. Pengecualian ditangkap secara otomatis saat dibuat dari permintaan yang diinstrumentasi. Anotasi dan metadata berisi informasi tambahan yang Anda rekam menggunakan operasi yang disediakan oleh X-Ray SDK. Untuk menambahkan anotasi atau metadata ke segmen Anda, gunakan X-Ray SDK. Untuk informasi selengkapnya, lihat tautan khusus bahasa yang tercantum di bawah Menginstrumentasi aplikasi Anda dengan in. AWS X-Ray SDKs [Instrumentasi aplikasi Anda untuk AWS X-Ray](#)

Melihat detail subsegmen

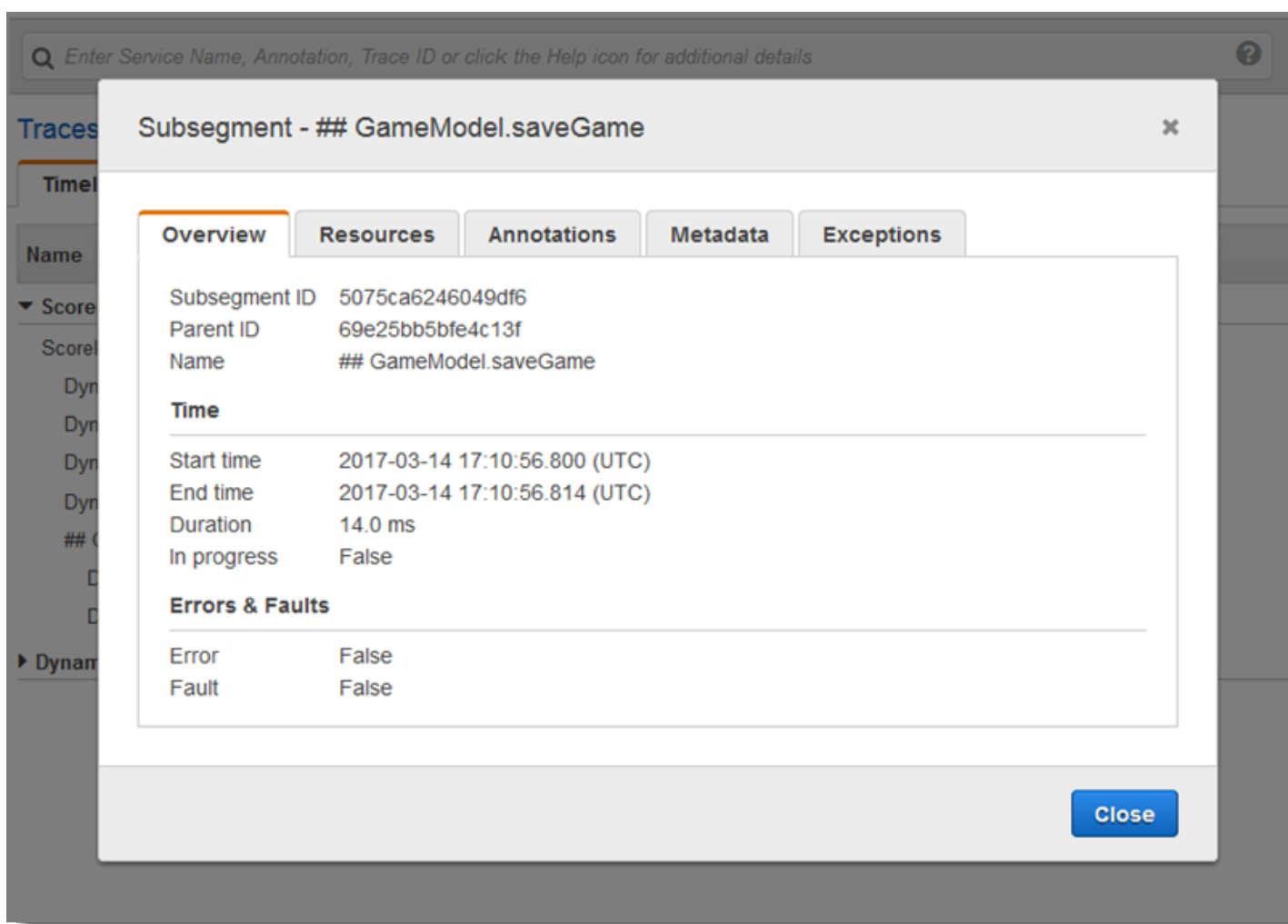
Dari timeline trace, pilih nama subsegmen untuk melihat detailnya:

- Tab Ikhtisar berisi informasi tentang permintaan dan respons. Ini termasuk nama, waktu mulai, waktu akhir, durasi, permintaan URL, operasi permintaan, kode respons permintaan, dan kesalahan dan kesalahan apa pun. Untuk subsegmen yang dihasilkan dengan klien yang diinstrumentasi, tab Gambaran Umum berisi informasi tentang permintaan dan respons dari sudut pandang aplikasi Anda.
- Tab Sumber Daya untuk subsegmen menunjukkan detail tentang AWS sumber daya yang digunakan untuk menjalankan subsegmen. Misalnya, tab sumber daya dapat mencakup AWS Lambda fungsi ARN, informasi tentang tabel DynamoDB, operasi apa pun yang dipanggil, dan ID permintaan.
- Tab yang tersisa menampilkan Anotasi, Metadata, dan Pengecualian yang direkam di subsegmen. Pengecualian ditangkap secara otomatis saat dibuat dari permintaan yang diinstrumentasi. Anotasi dan metadata berisi informasi tambahan yang Anda rekam menggunakan operasi yang disediakan oleh X-Ray SDK. Gunakan X-Ray SDK untuk menambahkan anotasi atau metadata ke segmen

Anda. Untuk informasi selengkapnya, lihat tautan khusus bahasa yang tercantum di bawah Menginstrumentasi aplikasi Anda dengan in. AWS X-Ray SDKs [Instrumentasi aplikasi Anda untuk AWS X-Ray](#)

Untuk subsegment kustom, tab Gambaran Umum menunjukkan nama subsegmen, yang dapat Anda atur untuk menentukan area kode atau fungsi yang dicatat. Untuk informasi selengkapnya, lihat tautan khusus bahasa yang tercantum di bawah Menginstrumentasi aplikasi Anda dengan in. AWS X-Ray SDKs [Membuat subsegment kustom dengan X-Ray SDK for Java](#)

Gambar berikut menunjukkan tab Ikhtisar untuk subsegment kustom. Ikhtisar berisi ID subsegment, ID induk, Nama, waktu mulai dan akhir, durasi, status, dan kesalahan atau kesalahan.



Tab Metadata untuk subsegment kustom berisi informasi JSON format tentang sumber daya yang digunakan oleh subsegment itu.

Menggunakan ekspresi filter

Gunakan ekspresi filter untuk melihat peta jejak atau jejak untuk permintaan, layanan, koneksi tertentu antara dua layanan (tepi), atau permintaan yang memenuhi suatu kondisi. X-Ray menyediakan bahasa ekspresi filter untuk memfilter permintaan, layanan, dan edge berdasarkan data dalam header permintaan, status respons, dan bidang yang diindeks pada segmen asli.

Ketika Anda memilih jangka waktu pelacakan untuk dilihat di konsol X-Ray, Anda mungkin mendapatkan lebih banyak hasil daripada yang dapat ditampilkan konsol tersebut. Di sudut kanan atas, konsol menunjukkan jumlah pelacakan yang dipindai dan apakah ada lebih banyak pelacakan yang tersedia. Anda dapat menggunakan ekspresi filter untuk mempersempit hasil hanya jejak yang ingin Anda temukan.

Topik

- [Detail ekspresi filter](#)
- [Menggunakan ekspresi filter dengan grup](#)
- [Sintaks ekspresi filter](#)
- [Kata Kunci Boolean](#)
- [Kata Kunci nomor](#)
- [Kata Kunci String](#)
- [Kata Kunci Kompleks](#)
- [fungsi id](#)

Detail ekspresi filter

Saat Anda [memilih node di peta jejak](#), konsol akan membuat ekspresi filter berdasarkan nama layanan node, dan jenis kesalahan yang ada berdasarkan pilihan Anda. Untuk menemukan pelacakan yang menunjukkan masalah performa atau yang terkait dengan permintaan tertentu, Anda dapat menyesuaikan ekspresi yang disediakan konsol tersebut atau membuat ekspresi Anda sendiri. Jika Anda menambahkan anotasi dengan X-Ray SDK, Anda juga dapat mem-filter berdasarkan keberadaan kunci anotasi atau nilai kunci.

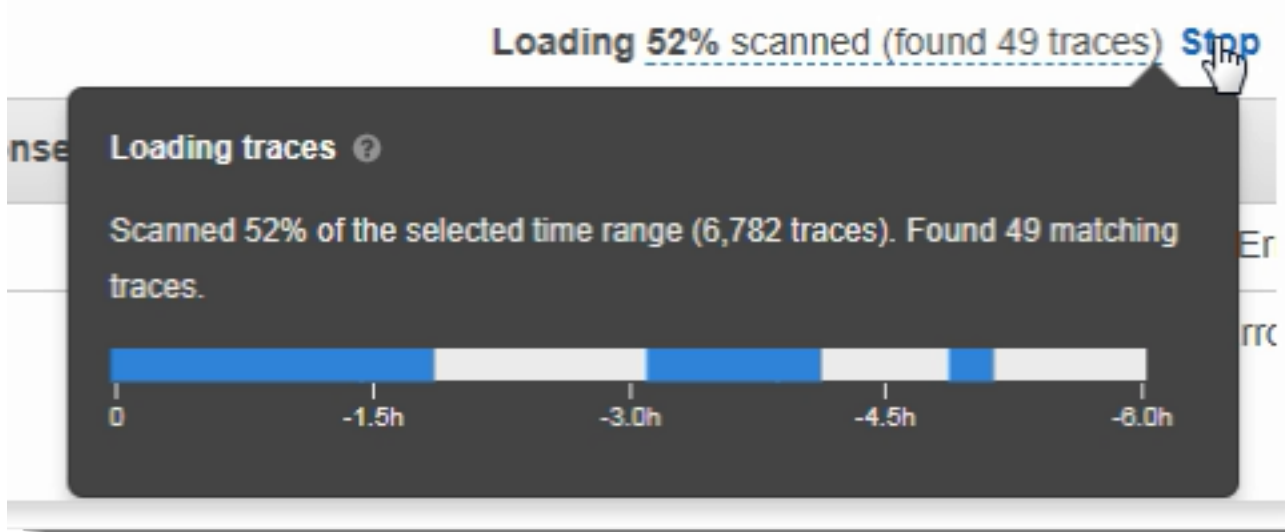
Note

Jika Anda memilih rentang waktu relatif di peta jejak dan memilih simpul, konsol mengubah rentang waktu menjadi waktu mulai dan berakhir mutlak. Untuk memastikan bahwa

pelacakan untuk simpul muncul di hasil pencarian, dan menghindari waktu pemindaian saat simpul tidak aktif, rentang waktu hanya mencakup waktu ketika simpul mengirim pelacakan. Untuk mencari relatif terhadap waktu saat ini, Anda dapat beralih kembali ke rentang waktu relatif di halaman pelacakan dan memindai kembali.

Jika masih ada lebih banyak hasil yang tersedia daripada yang dapat ditampilkan konsol, konsol menunjukkan kepada Anda berapa banyak pelacakan yang cocok dan jumlah pelacakan yang dipindai. Persentase yang ditampilkan adalah persentase dari kerangka waktu yang dipilih yang dipindai. Untuk memastikan bahwa Anda melihat semua pelacakan yang cocok terwakili dalam hasil, persempit ekspresi filter Anda lebih lanjut, atau pilih jangka waktu yang lebih singkat.

Untuk mendapatkan hasil terbaru terlebih dahulu, konsol mulai memindai pada akhir rentang waktu dan bekerja mundur. Jika ada banyak pelacakan, tetapi sedikit hasil, konsol tersebut membagi rentang waktu menjadi beberapa bagian dan memindainya secara paralel. Bilah kemajuan menunjukkan bagian dari rentang waktu yang telah dipindai.



Menggunakan ekspresi filter dengan grup

Grup adalah kumpulan pelacakan yang ditentukan oleh ekspresi filter. Anda dapat menggunakan grup untuk menghasilkan grafik layanan tambahan dan menyediakan CloudWatch metrik Amazon.

Grup diidentifikasi berdasarkan namanya atau Amazon Resource Name (ARN), dan berisi ekspresi filter. Layanan membandingkan pelacakan yang masuk dengan ekspresi dan menyimpannya sesuai dengan itu.

Anda dapat membuat dan memodifikasi grup dengan menggunakan menu tarik-turun di sebelah kiri bilah pencarian ekspresi filter.

Note

Jika layanan mengalami kesalahan dalam kualifikasi grup, grup tersebut tidak lagi disertakan dalam memproses pelacakan masuk dan metrik kesalahan dicatat.

Untuk informasi selengkapnya tentang grup opsi, lihat [Mengkonfigurasi grup](#).

Sintaks ekspresi filter

Ekspresi filter dapat berisi Kata Kunci, unary atau biner Operator, dan nilai untuk perbandingan.

keyword operator value

Operator yang berbeda tersedia untuk berbagai tipe kata kunci. Misalnya, `responsetime` adalah kata kunci nomor dan dapat dibandingkan dengan operator yang terkait dengan nomor.

Example – permintaan saat waktu respons lebih besar dari 5 detik

```
responsetime > 5
```

Anda dapat menggabungkan beberapa ekspresi dalam ekspresi gabungan dengan menggunakan operator AND atau OR.

Example – permintaan saat total durasi adalah 5-8 detik

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

Kata kunci dan operator sederhana hanya menemukan masalah di tingkat penelusuran. Jika kesalahan terjadi di hilir, tetapi ditangani oleh aplikasi Anda dan tidak dikembalikan ke pengguna, pencarian untuk `error` tidak akan menemukannya.

Untuk menemukan pelacakan dengan masalah hilir, Anda dapat menggunakan [kata kunci kompleks](#) `service()` dan `edge()`. Kata kunci ini memungkinkan Anda menerapkan ekspresi filter ke semua simpul hilir, satu simpul hilir, atau tepi di antara dua simpul. Untuk perincian lebih lanjut, Anda dapat memfilter layanan dan edge berdasarkan tipe dengan [fungsi `id\(\)`](#).

Kata Kunci Boolean

Nilai kata kunci Boolean adalah BETUL atau SALAH. Gunakan kata kunci ini untuk menemukan pelacakan yang mengakibatkan kesalahan.

Kata Kunci Boolean

- `ok` – Kode status respons 2XX Berhasil.
- `error` – Kode status respons 4XX Kesalahan Klien.
- `throttle` – Kode status respons 429 Terlalu Banyak Permintaan.
- `fault` – Kode status respons 5XX Kesalahan Server.
- `partial` – Permintaan memiliki segmen yang tidak lengkap.
- `inferred` – Permintaan memiliki segmen yang disimpulkan.
- `first` - Elemen adalah yang pertama dari daftar enumerasi.
- `last` - Elemen adalah yang terakhir dari daftar enumerasi.
- `remote` – Entitas akar masalah jarak jauh.
- `root` – Layanan adalah titik masuk atau segmen akar dari suatu pelacakan.

Operator Boolean menemukan segmen saat kunci yang ditentukan adalah `true` atau `false`.

Operator Boolean

- Tidak ada - Ekspresi benar jika kata kunci benar.
- `!` - Ekspresi benar jika kata kunci salah.
- `=, !=` – Bandingkan nilai kata kunci dengan string `true` atau `false`. Operator ini bertindak sama dengan operator lain tetapi lebih eksplisit.

Example – status respons 2XX OK

```
ok
```

Example – status respons bukan 2XX OK

```
!ok
```

Example – status respons bukan 2XX OK

```
ok = false
```

Example – pelacakan kesalahan yang disebutkan terakhir memiliki nama kesalahan "deserialize"

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

Example – permintaan dengan segmen jarak jauh saat cakupan lebih besar dari 0,7 dan nama layanannya adalah "pelacakan"

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

Example – permintaan dengan segmen yang disimpulkan dengan tipe layanan "AWS:DynamoDB"

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

Example – permintaan yang memiliki segmen dengan nama "data-plane" sebagai root

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

Kata Kunci nomor

Gunakan kata kunci angka untuk mencari permintaan dengan waktu respons, durasi, atau status respons tertentu.

Kata Kunci nomor

- `responsetime` – Waktu yang dibutuhkan server untuk mengirim respons.
- `duration` – Total permintaan, termasuk semua panggilan hilir.
- `http.status` – Kode status respons.
- `index` – Posisi elemen dalam daftar enumerasi.
- `coverage` – Persentase desimal waktu respons entitas selama waktu respons segmen akar. Hanya berlaku untuk entitas akar masalah waktu respons.

Operasi nomor

Kata kunci angka menggunakan persamaan standar dan operator perbandingan.

- `=,!=` – Kata kunci sama dengan atau tidak sama dengan nilai angka.
- `<,<=,>,>=` – Kata kunci kurang dari atau lebih besar dari nilai angka.

Example – status respons bukan 200 OK

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

Example – permintaan saat total durasinya adalah 5–8 detik

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

Example – permintaan yang berhasil diselesaikan dalam waktu kurang dari 3 detik, termasuk semua panggilan hilir

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

Example – entitas daftar enumerasi yang memiliki indeks lebih besar dari 5

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

Example – permintaan saat entitas terakhir yang memiliki cakupan lebih besar dari 0,8

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

Kata Kunci String

Gunakan kata kunci string untuk menemukan jejak dengan teks tertentu di header permintaan, atau pengguna IDs tertentu.

Kata Kunci String

- `http.url` – Meminta URL.
- `http.method` – Metode permintaan.
- `http.useragent` – Meminta string agen pengguna.
- `http.clientip` – Alamat IP Peminta.
- `user` – Nilai dari bidang pengguna pada setiap segmen dalam pelacakan.
- `name` – Nama layanan atau pengecualian.

- `type` – Tipe Layanan.
- `message` – Pesan pengecualian.
- `availabilityzone` – Nilai ketersediaan bidang zona pada setiap segmen dalam pelacakan.
- `instance.id` – Nilai bidang ID instans pada setiap segmen dalam pelacakan.
- `resource.arn` – Nilai bidang ARN sumber daya pada setiap segmen dalam pelacakan.

Operator string menemukan nilai yang sama dengan atau berisi teks tertentu. Nilai harus selalu ditentukan dalam tanda kutip.

Operator String

- `=, !=` – Kata kunci adalah sama dengan atau tidak sama dengan nilai angka.
- `CONTAINS` – Kata kunci berisi string tertentu.
- `BEGINSWITH, ENDSWITH` – Kata kunci dimulai atau diakhiri dengan string tertentu.

Example – filter `http.url`

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

Untuk menguji apakah suatu bidang ada pada pelacakan, terlepas dari nilainya, periksa untuk melihat apakah bidang tersebut berisi string kosong.

Example – filter pengguna

Temukan semua jejak dengan pengguna IDs.

```
user CONTAINS ""
```

Example – pilih pelacakan dengan penyebab akar masalah yang mencakup layanan bernama "Auth"

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

Example – pilih pelacakan dengan akar masalah waktu respons yang layanan terakhirnya memiliki tipe DynamoDB

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

Example – pilih pelacakan dengan akar penyebab kesalahan yang pengecualian terakhirnya memiliki pesan "akses ditolak untuk akun_id: 1234567890"

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

Kata Kunci Kompleks

Gunakan kata kunci yang kompleks untuk menemukan permintaan berdasarkan nama layanan, nama edge, atau nilai anotasi. Untuk layanan dan edge, Anda dapat menentukan ekspresi filter tambahan yang berlaku untuk layanan atau edge. Untuk anotasi, Anda dapat memfilter nilai anotasi dengan kunci tertentu menggunakan operator Boolean, angka, atau string.

Kata Kunci Kompleks

- `annotation[key]`— Nilai anotasi dengan bidang *key*. Nilai anotasi dapat menjadi Boolean, nomor, atau string, sehingga Anda dapat menggunakan salah satu operator perbandingan tipe tersebut. Anda dapat menggunakan kata kunci ini dalam kombinasi dengan edge kata kunci `service` atau. Kunci anotasi yang berisi titik (titik) harus dibungkus dengan tanda kurung siku ([]).
- `edge(source, destination) {filter}`— Koneksi antara layanan *source* dan *destination*. Kawat lengkung opsional dapat berisi ekspresi filter yang berlaku untuk segmen pada koneksi ini.
- `group.name / group.arn` – Nilai ekspresi filter grup, yang dirujuk oleh nama grup atau ARN grup.
- `json` – objek akar masalah JSON. Lihat [Mendapatkan data dari AWS X-Ray](#) untuk langkah-langkah membuat entitas JSON secara terprogram.
- `service(name) {filter}`— Layanan dengan nama *name*. Kawat lengkung opsional dapat berisi ekspresi filter yang berlaku untuk segmen yang dibuat oleh layanan.

Gunakan kata kunci layanan untuk menemukan jejak permintaan yang mengenai node tertentu di peta jejak Anda.

Operator kata kunci yang kompleks menemukan segmen saat kunci yang ditentukan telah diset, atau tidak diset.

Operator kata kunci kompleks

- Tidak ada - Ekspresi benar jika kata kunci diset. Jika kata kunci bertipe boolean, maka akan dievaluasi ke nilai boolean.
- ! – Ekspresi benar jika kata kunci tidak diset. Jika kata kunci bertipe boolean, maka akan dievaluasi ke nilai boolean.
- =, != – Bandingkan nilai kata kunci.
- edge(*source*, *destination*) {*filter*}— Koneksi antara layanan *source* dan *destination*. Kawat lengkung opsional dapat berisi ekspresi filter yang berlaku untuk segmen pada koneksi ini.
- annotation[*key*]— Nilai anotasi dengan bidang *key*. Nilai anotasi dapat menjadi Boolean, nomor, atau string, sehingga Anda dapat menggunakan salah satu operator perbandingan tipe tersebut. Anda dapat menggunakan kata kunci ini dalam kombinasi dengan edge kata kunci service atau.
- json – objek akar masalah JSON. Lihat [Mendapatkan data dari AWS X-Ray](#) untuk langkah-langkah membuat entitas JSON secara terprogram.

Gunakan kata kunci layanan untuk menemukan jejak permintaan yang mengenai node tertentu di peta jejak Anda.

Example – Filter layanan

Permintaan yang mencakup panggilan ke `api.example.com` dengan kesalahan (kesalahan seri 500).

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

Anda dapat mengecualikan nama layanan untuk menerapkan ekspresi filter ke semua simpul di peta layanan Anda.

Example – Filter layanan

Permintaan yang menyebabkan kesalahan di mana saja di peta jejak Anda.

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

Kata kunci edge menerapkan ekspresi filter ke koneksi antara dua simpul.

Example – filter edge

Permintaan saat layanan `api.example.com` melakukan panggilan ke `backend.example.com` yang gagal dengan kesalahan.

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

Anda juga dapat menggunakan operator `!` dengan kata kunci layanan dan edge untuk mengecualikan layanan atau edge dari hasil ekspresi filter lain.

Example – filter layanan dan permintaan

Permintaan saat URL dimulai dengan `http://api.example.com/` dan berisi `/v2/` tetapi tidak menjangkau layanan bernama `api.example.com`.

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

Example – filter layanan dan waktu respons

Temukan pelacakan saat `http url` diset dan waktu respons lebih dari 2 detik.

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

Untuk anotasi, Anda dapat menghubungkan semua pelacakan tempat `annotation[key]` diatur, atau menggunakan operator perbandingan yang sesuai dengan tipe nilai.

Example – anotasi dengan nilai string

Permintaan dengan anotasi bernama `gameid` dengan nilai string `"817DL6V0"`.

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

Example – anotasi diset

Permintaan dengan anotasi bernama `set age`.

```
annotation[age]
```

Example – anotasi tidak diset

Permintaan tanpa anotasi bernama `set age`.


```
!annotation[age]
```

Example – anotasi dengan nilai angka

Permintaan dengan usia anotasi dengan nilai numerik lebih besar dari 29.

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

Example — anotasi dalam kombinasi dengan layanan atau tepi

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

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

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

Example – grup dengan pengguna

Permintaan saat pelacakan memenuhi filter grup `high_response_time` (misalnya `responseTime > 3`), dan pengguna bernama Alice.

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

Example – JSON dengan akar masalah entitas

Permintaan dengan entitas akar masalah yang cocok.

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

fungsi id

Ketika Anda memberikan nama layanan untuk kata kunci `service` atau `edge`, Anda mendapatkan hasil untuk semua simpul yang memiliki nama tersebut. Untuk pemfilteran yang lebih tepat, Anda dapat menggunakan fungsi `id` untuk menentukan tipe layanan selain nama untuk membedakan antara simpul dengan nama yang sama.

Gunakan `account.id` fungsi untuk menentukan akun tertentu untuk layanan, saat melihat jejak dari beberapa akun di akun pemantauan.

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

Anda dapat menggunakan fungsi `id` sebagai pengganti nama layanan dalam filter layanan dan 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 }
```

Misalnya, AWS Lambda fungsi menghasilkan dua node di peta jejak; satu untuk pemanggilan fungsi, dan satu untuk layanan Lambda. Kedua simpul memiliki nama yang sama tetapi berbeda tipe. Filter layanan standar akan menemukan pelacakan untuk keduanya.

Example – Filter layanan

Permintaan yang mencakup kesalahan pada layanan bernama `random-name`.

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

Gunakan fungsi `id` untuk mempersempit pencarian ke kesalahan pada fungsi itu sendiri, tidak termasuk kesalahan dari layanan.

Example – filter layanan dengan fungsi id

Permintaan yang mencakup kesalahan pada layanan bernama `random-name` dengan tipe `AWS::Lambda::Function`.

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

Untuk mencari simpul berdasarkan tipe, Anda juga dapat mengecualikan nama sepenuhnya.

Example — filter layanan dengan fungsi id dan jenis layanan

Permintaan yang menyertakan kesalahan pada layanan dengan tipe `AWS::Lambda::Function`.

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

Untuk mencari node tertentu Akun AWS, tentukan ID akun.

Example — filter layanan dengan fungsi id dan ID akun

Permintaan yang menyertakan layanan dalam ID akun tertentu `AWS::Lambda::Function`.

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

Penelusuran lintas akun

AWS X-Ray mendukung pengamatan lintas akun, memungkinkan Anda untuk memantau dan memecahkan masalah aplikasi yang menjangkau beberapa akun dalam file. Wilayah AWS Anda dapat dengan mulus mencari, memvisualisasikan, dan menganalisis metrik, log, dan jejak Anda di salah satu akun tertaut seolah-olah Anda beroperasi dalam satu akun. Ini memberikan tampilan lengkap permintaan yang melintasi beberapa akun. Anda dapat melihat jejak lintas akun di peta jejak X-Ray dan melacak halaman di dalam [CloudWatch konsol](#).

Data observabilitas bersama dapat mencakup salah satu jenis telemetry berikut:

- Metrik di Amazon CloudWatch
- Grup log di Amazon CloudWatch Logs
- Jejak di AWS X-Ray
- Aplikasi dalam Wawasan CloudWatch Aplikasi Amazon

Konfigurasi observabilitas lintas akun

Untuk mengaktifkan observabilitas lintas akun, siapkan satu atau beberapa akun AWS pemantauan dan tautkan dengan beberapa akun sumber. Akun pemantauan adalah pusat Akun AWS yang dapat melihat dan berinteraksi dengan data observabilitas yang dihasilkan dari akun sumber. Akun sumber adalah individu Akun AWS yang menghasilkan data observabilitas untuk sumber daya yang dikandungnya.

Akun sumber membagikan data observabilitas mereka dengan akun pemantauan. Jejak disalin dari setiap akun sumber hingga lima akun pemantauan. Salinan jejak dari akun sumber ke akun pemantauan pertama gratis. Salinan jejak yang dikirim ke akun pemantauan tambahan dibebankan ke setiap akun sumber, berdasarkan harga standar. Untuk informasi selengkapnya, lihat [AWS X-Ray harga](#) dan [CloudWatch harga Amazon](#).

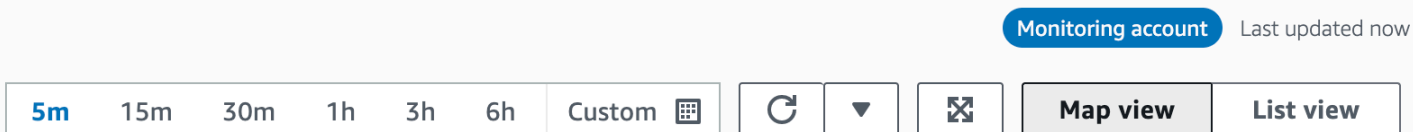
Untuk membuat tautan antara akun pemantauan dan akun sumber, gunakan CloudWatch konsol atau perintah Observability Access Manager baru di API AWS CLI dan. Untuk informasi lebih lanjut, lihat [CloudWatch observabilitas lintas akun](#).

Note

Jejak X-Ray ditagih ke Akun AWS tempat mereka diterima. Jika permintaan [sampel](#) mencakup layanan di lebih dari satu Akun AWS, setiap akun akan mencatat jejak terpisah, dan semua jejak berbagi ID jejak yang sama. Untuk mempelajari lebih lanjut tentang harga observabilitas lintas akun, lihat [AWS X-Ray harga dan harga Amazon CloudWatch](#).

Melihat jejak lintas akun

Jejak lintas akun ditampilkan di akun pemantauan. Setiap akun sumber hanya menampilkan jejak lokal untuk akun tertentu. Bagian berikut mengasumsikan bahwa Anda masuk ke akun pemantauan dan telah membuka CloudWatch konsol Amazon. Pada peta jejak dan halaman jejak, rencana akun pemantauan ditampilkan di sudut kanan atas.



Peta jejak

Di CloudWatch konsol, pilih Trace Map di bawah jejak X-Ray dari panel navigasi kiri. Secara default, peta jejak menampilkan node untuk semua akun sumber yang mengirim jejak ke akun pemantauan, dan node untuk akun pemantauan itu sendiri. Pada peta jejak, pilih Filter dari kiri atas untuk memfilter peta jejak menggunakan drop-down Akun. Setelah filter akun diterapkan, node layanan dari akun yang tidak cocok dengan filter saat ini akan berwarna abu-abu.

The screenshot shows the AWS X-Ray console interface. At the top, there is a 'Filters' button with a count of 1. Below it, a search bar says 'Filter by X-Ray group'. To the right, there is a '+' button and another search bar that says 'Select a node'. A 'Filters' modal is open, showing 'Accounts' with a dropdown menu set to 'Select accounts'. Below the dropdown, there is a list of accounts with 'Monitoring account 1234567890123' selected. There is a 'Clear all' button. At the bottom of the modal, there is a 'Map layout' section with a toggle switch for 'Show nodes with no filters applied.' The background shows a Lambda function node labeled 'TestLambda' with a status of 'Ok 100% 1.00 t/min'.

Saat Anda memilih node layanan, panel rincian node menyertakan ID dan label akun layanan.

The screenshot shows the details of the 'TestLambda' node. At the top, there is a dropdown menu set to 'TestLambda'. Below it, there is a section for 'Lambda Function' with the account ID 'Monitoring account (1234567890123)'. To the right of this section, there are four buttons: 'View logs', 'View traces', 'Analyze traces', and 'View dashboard'. Below this, there is a section for 'Metrics' with tabs for 'Metrics', 'Alerts', and 'Response time distribution'.

Di sudut kanan atas peta jejak, pilih Tampilan daftar untuk melihat daftar node layanan. Daftar node layanan mencakup layanan dari akun pemantauan dan semua akun sumber yang dikonfigurasi. Filter daftar node berdasarkan label Akun atau ID Akun dengan memilihnya dari filter Node.

The screenshot shows the 'Nodes (2)' section of the AWS X-Ray console. At the top, there is a search bar with the text 'Account id ='. Below the search bar, there is a dropdown menu with the text 'Use: "Account id = "'. To the right of the search bar, there is a table with columns for 'Alarms', 'Latency (avg)', and 'Faults (5xx)'. The table has one row with the following values: 'Account id = 461265027466', '1', '13ms', and '0.00/min'.

Pelacakan

Lihat detail jejak untuk jejak yang menjangkau beberapa akun dengan membuka CloudWatch konsol dari akun pemantauan, dan memilih Jejak di bawah jejak X-Ray di panel navigasi kiri. Anda juga dapat membuka halaman ini dengan memilih node di X-Ray Trace Map, dan kemudian memilih Lihat jejak dari panel rincian node.

Halaman Jejak mendukung kueri berdasarkan ID akun. Untuk memulai, [masukkan kueri yang](#) menyertakan satu atau beberapa akun IDs. Contoh kueri berikut untuk jejak yang telah melewati ID akun X atau Y:

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

The screenshot shows the AWS X-Ray Traces console interface. At the top, there's a 'Traces Info' header with a time range selector set to '5m'. Below this, a text box contains the query `service(id(account.id: "1234567890123"))`. To the left of the query is a search icon and the text 'Filter by X-Ray group'. Below the query input is an orange 'Run query' button and a green status indicator that says '5 traces retrieved'.

Perbaiki kueri Anda berdasarkan Akun. Pilih satu atau beberapa akun dari daftar, dan pilih Tambahkan ke kueri.

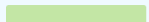
The screenshot shows the 'Query refiners' section in the AWS X-Ray console. It has a dropdown menu labeled 'Refine query by' with 'Account' selected. To the right, it says '1 selected' and there is an orange 'Add to query' button. Below this is a search bar with the placeholder text 'Find Account name and ID'. At the bottom, there is a list of refiners. The first one is 'Account name and ID' with a checkbox checked. The second one is 'Monitoring account (1234567890123)' with a checkbox checked and the entire row highlighted in light blue.

Detail jejak

Lihat detail untuk jejak dengan memilihnya dari daftar Jejak di bagian bawah halaman Jejak. Detail Trace ditampilkan, termasuk peta detail jejak dengan node layanan dari seluruh akun yang dilalui jejak. Pilih node layanan tertentu untuk melihat akun yang sesuai.

Bagian timeline Segments menampilkan detail akun untuk setiap segmen di timeline.

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

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

Menelusuri aplikasi berbasis peristiwa

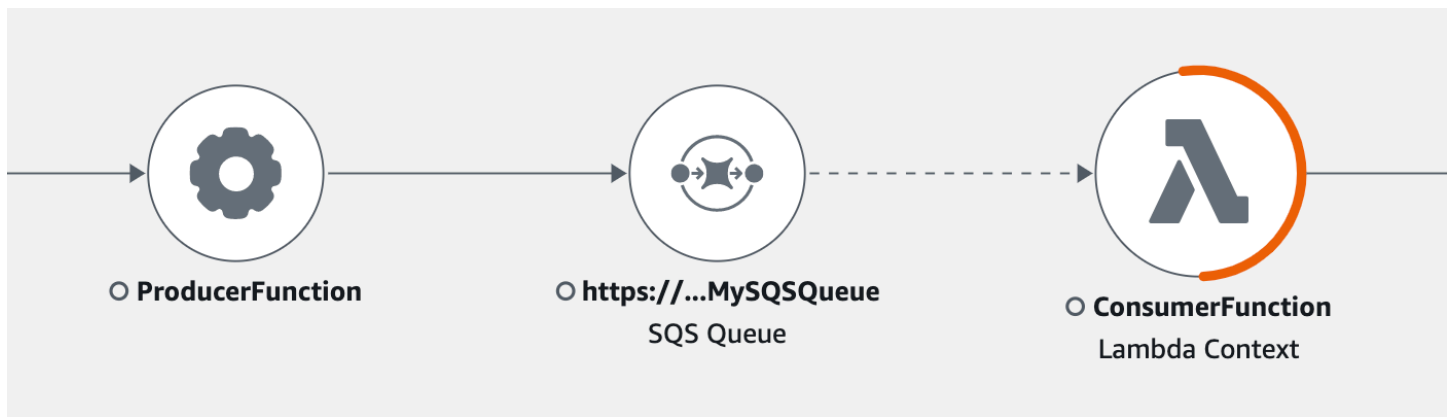
AWS X-Ray mendukung penelusuran aplikasi berbasis peristiwa menggunakan Amazon SQS dan AWS Lambda. Gunakan CloudWatch konsol untuk melihat tampilan tersambung dari setiap permintaan saat diantri dengan Amazon SQS dan diproses oleh satu atau beberapa fungsi Lambda. Jejak dari produsen pesan hulu secara otomatis ditautkan ke jejak dari node konsumen Lambda hilir, menciptakan end-to-end tampilan aplikasi.

Note

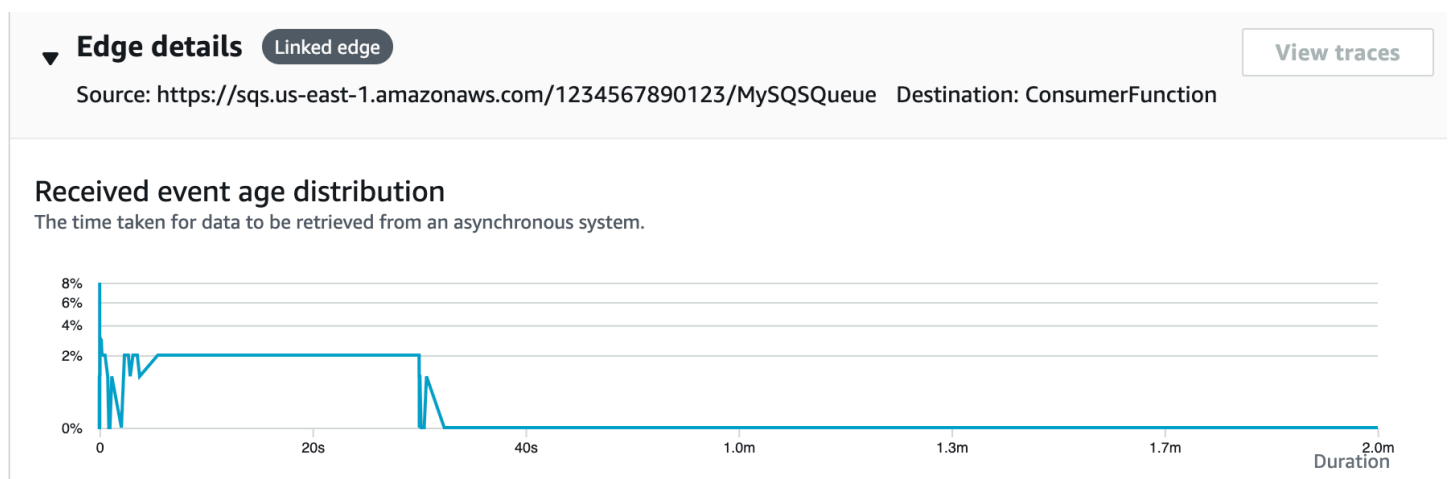
Setiap segmen jejak dapat ditautkan hingga 20 jejak, sementara jejak dapat mencakup maksimum 100 tautan. Dalam skenario tertentu, menautkan jejak tambahan dapat mengakibatkan melebihi [ukuran dokumen jejak maksimum](#), menyebabkan jejak yang berpotensi tidak lengkap. Ini dapat terjadi, misalnya, ketika fungsi Lambda dengan penelusuran diaktifkan mengirim banyak pesan SQS ke antrian dalam satu pemanggilan. Jika Anda mengalami masalah ini, mitigasi tersedia yang menggunakan X-Ray. SDKs Lihat X-Ray SDK untuk [Java](#), [Node.js](#), [Python](#), [Go](#), [atau](#) [.NET](#) untuk informasi lebih lanjut.

Lihat jejak yang ditautkan di peta jejak

Gunakan halaman Trace Map dalam [CloudWatch konsol](#) untuk melihat peta jejak dengan jejak dari produsen pesan yang ditautkan ke jejak dari konsumen Lambda. Tautan ini ditampilkan dengan tepi garis putus-putus yang menghubungkan node Amazon SQS dan node konsumen Lambda hilir.



Pilih tepi garis putus-putus untuk menampilkan histogram usia acara yang diterima, yang memetakan penyebaran usia acara saat diterima oleh konsumen. Usia dihitung setiap kali suatu acara diterima.



Lihat detail jejak yang ditautkan

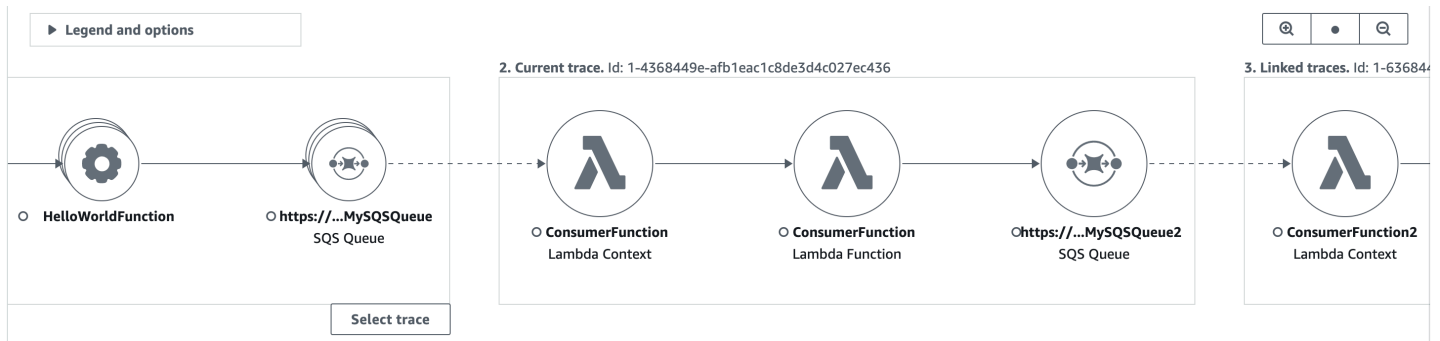
Melihat detail jejak yang dikirim dari produsen pesan, antrean Amazon SQS, atau konsumen Lambda:

1. Gunakan Trace Map untuk memilih produsen pesan, Amazon SQS, atau node konsumen Lambda.
2. Pilih Lihat jejak dari panel detail simpul untuk menampilkan daftar jejak. Anda juga dapat menavigasi langsung ke halaman Jejak di dalam CloudWatch konsol.
3. Pilih jejak tertentu dari daftar untuk membuka halaman detail jejak. Halaman detail jejak menampilkan pesan saat jejak yang dipilih adalah bagian dari kumpulan jejak yang ditautkan.

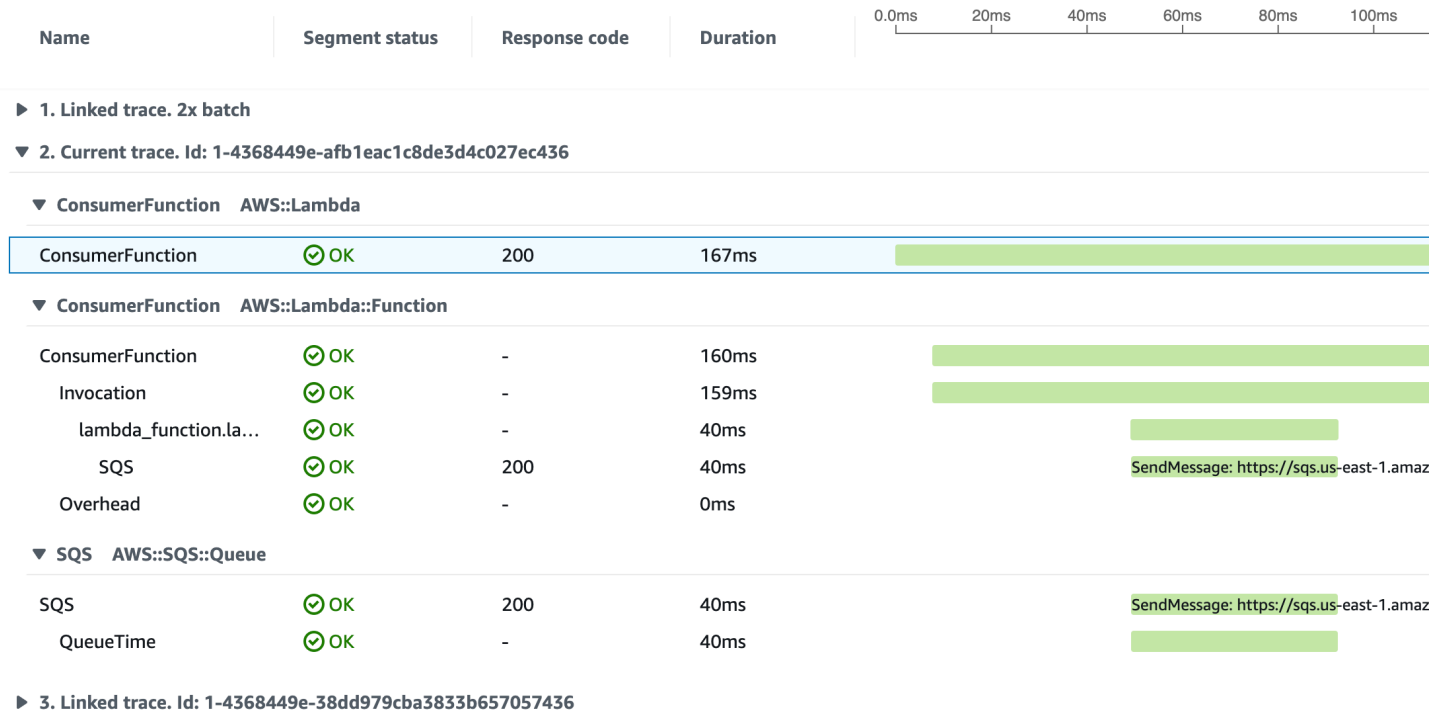
[CloudWatch](#) > [Traces](#) > Trace 1-4368449e-afb1eac1c8de3d4c027ec436**Trace 1-6368449e-afb1eac1c8de3d4c027ec436**[Info](#)

This trace is part of a linked set of traces

Peta detail jejak menampilkan jejak saat ini, bersama dengan jejak terkait hulu dan hilir, yang masing-masing terkandung dalam kotak yang menunjukkan batas setiap jejak. Jika jejak yang dipilih saat ini ditautkan ke beberapa jejak hulu atau hilir, node dalam jejak tertaut hulu atau hilir ditumpuk, dan tombol Select trace ditampilkan.



Di bawah peta detail jejak, garis waktu segmen jejak ditampilkan, termasuk jejak tertaut hulu dan hilir. Jika ada beberapa jejak tertaut hulu atau hilir, detail segmennya tidak dapat ditampilkan. Untuk melihat detail segmen untuk satu jejak dalam satu set jejak tertaut, [pilih satu jejak seperti yang](#) dijelaskan di bawah ini.

Segments Timeline [Info](#)

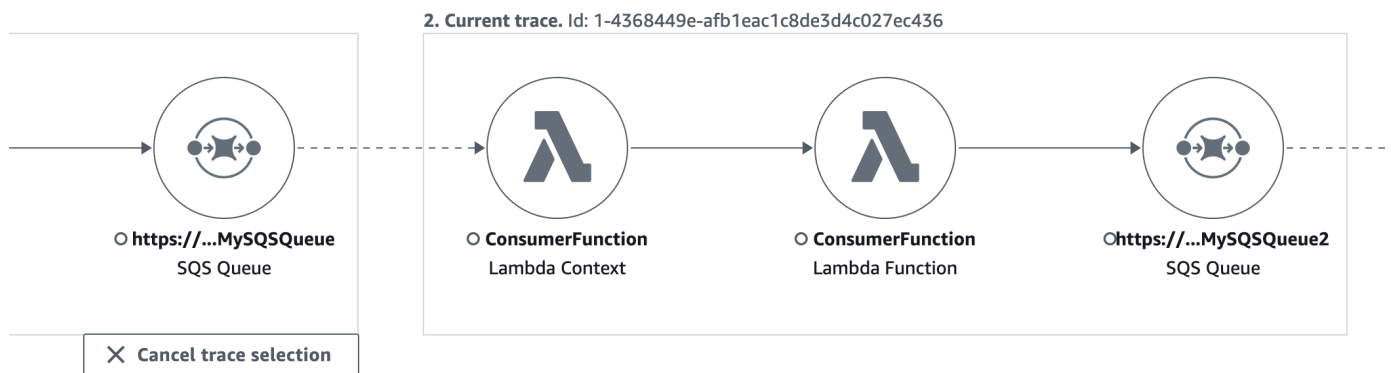
Pilih satu jejak dalam satu set jejak yang ditautkan

Filter kumpulan jejak yang ditautkan ke satu jejak, untuk melihat detail segmen di timeline.

1. Pilih jejak di bawah jejak yang ditautkan pada peta detail jejak. Daftar jejak ditampilkan.

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

2. Pilih tombol radio di sebelah jejak untuk melihatnya di dalam peta detail jejak.
3. Pilih Batalkan pilihan jejak untuk melihat seluruh rangkaian jejak yang ditautkan.



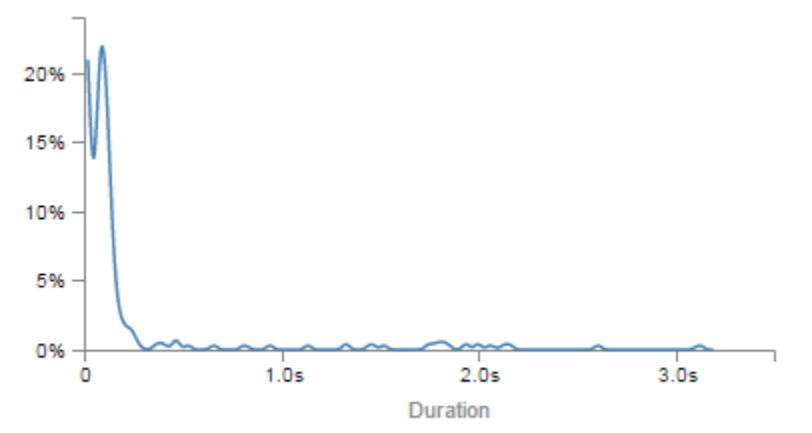
Menggunakan histogram latensi

Saat Anda memilih simpul atau tepi pada [peta AWS X-Ray jejak](#), konsol X-Ray menunjukkan histogram distribusi latensi.

Latensi

Latensi adalah jumlah waktu antara ketika permintaan dimulai dan ketika selesai. Histogram menunjukkan distribusi latensi. Hal tersebut menunjukkan durasi pada sumbu x, dan persentase permintaan yang cocok dengan setiap durasi pada sumbu y.

Histogram ini menunjukkan layanan yang melengkapi sebagian besar permintaan dalam waktu kurang dari 300 milidetik. Sebagian kecil permintaan memakan waktu hingga 2 detik, dan beberapa outlier membutuhkan lebih banyak waktu.



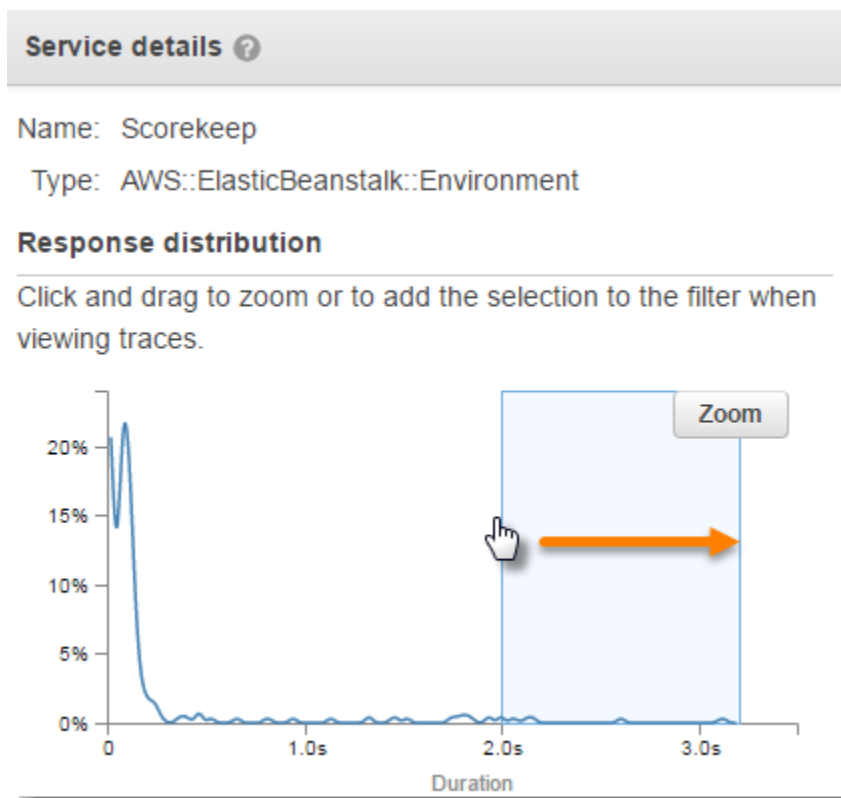
Menafsirkan detail layanan

Layanan histogram dan histogram edge memberikan representasi visual latensi dari sudut pandang layanan atau peminta.

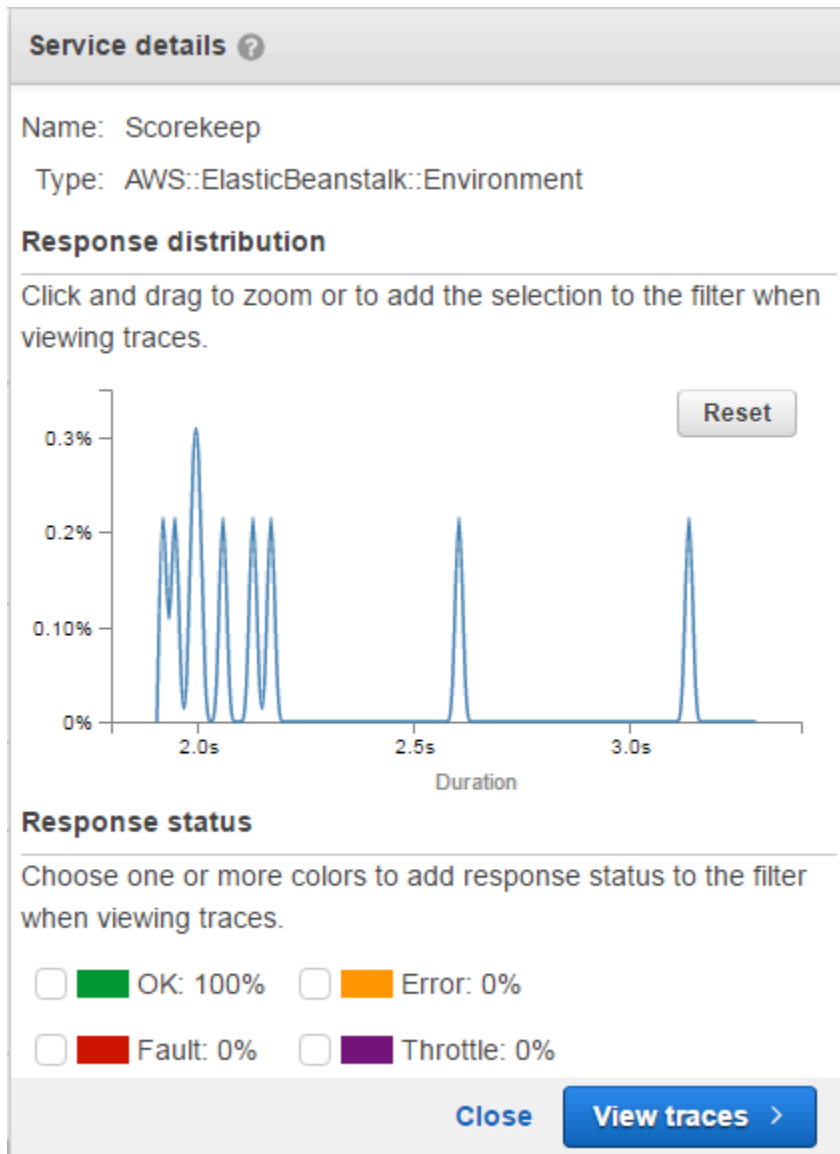
- Pilih Simpul layanan dengan mengeklik lingkaran. X-Ray menunjukkan histogram untuk permintaan yang dilayani oleh layanan. Latensi adalah yang dicatat oleh layanan, dan tidak menyertakan latensi jaringan apa pun antara layanan dan peminta.
- Pilih edge dengan mengeklik garis atau edge ujung panah antara dua layanan. X-Ray menunjukkan histogram untuk permintaan dari peminta yang dilayani oleh layanan downstream. Latensi adalah yang dicatat oleh peminta, dan termasuk latensi dalam koneksi jaringan antara dua layanan.

Untuk menafsirkan panel histogram Detail layanan, Anda dapat mencari nilai-nilai yang paling berbeda dari mayoritas nilai dalam histogram. Outlier ini dapat dilihat sebagai puncak atau lonjakan di histogram, dan Anda dapat melihat pelacakan untuk area tertentu untuk menyelidiki apa yang terjadi.

Untuk melihat pelacakan yang difilter menurut latensi, pilih rentang pada histogram. Klik di tempat Anda ingin memulai pemilihan dan seret dari kiri ke kanan untuk menyorot rentang latensi yang akan disertakan dalam filter pelacakan.



Setelah memilih rentang, Anda dapat memilih Perbesar untuk melihat bagian histogram itu dan menyempurnakan pilihan Anda.



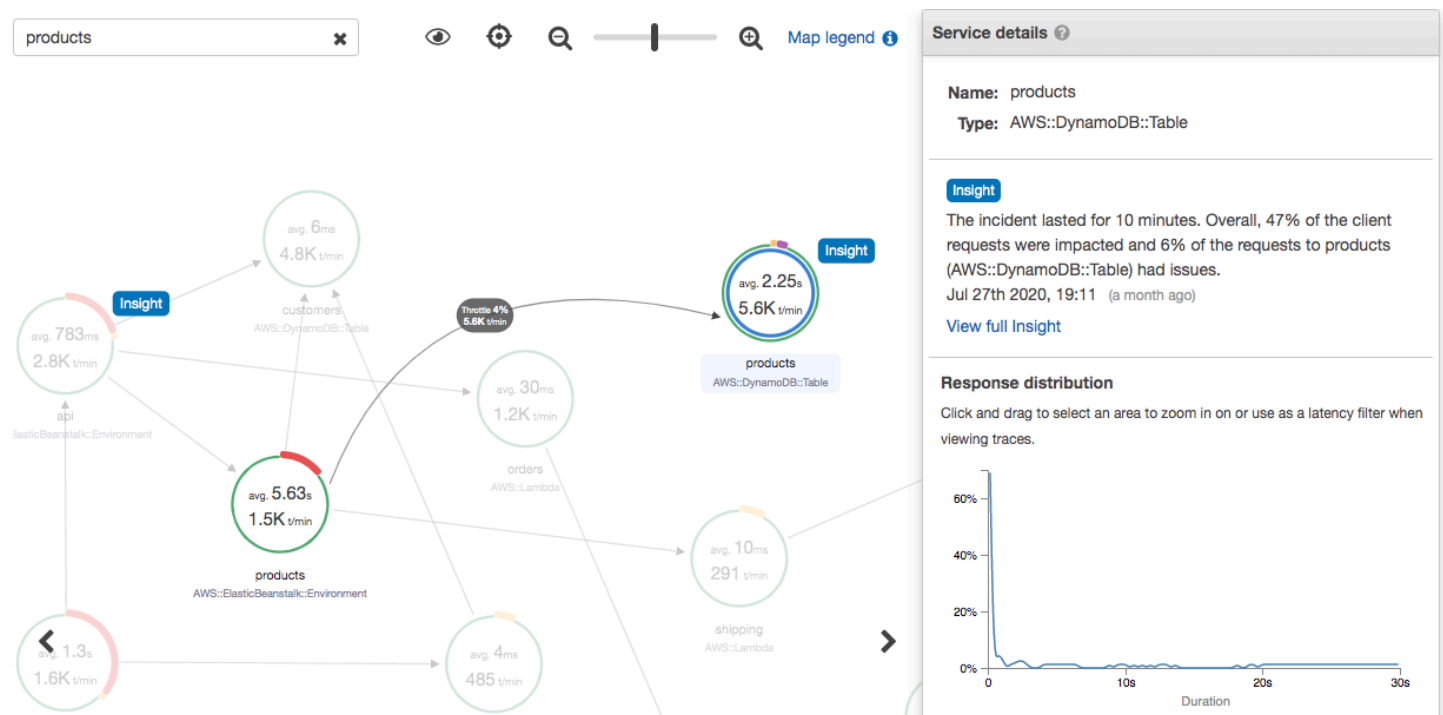
Setelah Anda mengatur fokus ke area yang Anda minati, pilih Lihat pelacakan.

Menggunakan wawasan X-Ray

AWS X-Ray terus menganalisis data jejak di akun Anda untuk mengidentifikasi masalah yang muncul dalam aplikasi Anda. Ketika tingkat kesalahan melebihi rentang yang diharapkan, itu menciptakan wawasan yang mencatat masalah dan melacak dampaknya hingga diselesaikan. Dengan wawasan, Anda dapat:

- Mengidentifikasi tempat terjadinya masalah di aplikasi Anda, akar penyebab masalah, dan dampak terkait. Analisis dampak yang diberikan oleh wawasan memungkinkan Anda memperoleh tingkat kepelikan dan prioritas suatu masalah.
- Menerima notifikasi saat masalah berubah seiring waktu. Pemberitahuan wawasan dapat diintegrasikan dengan solusi pemantauan dan peringatan Anda menggunakan Amazon EventBridge Integrasi ini memungkinkan Anda mengirim email atau peringatan otomatis berdasarkan tingkat keparahan masalah.

Konsol X-Ray mengidentifikasi node dengan insiden yang sedang berlangsung di peta jejak. Untuk melihat ringkasan wawasan, pilih simpul yang terpengaruh. Anda juga dapat melihat dan memfilter wawasan dengan memilih Wawasan dari panel navigasi di sebelah kiri.



X-Ray membuat wawasan saat mendeteksi anomali di satu atau beberapa simpul peta layanan. Layanan ini menggunakan pemodelan secara statistik untuk memprediksi tingkat kerusakan yang diharapkan dari layanan dalam aplikasi Anda. Dalam contoh sebelumnya, anomali adalah peningkatan kesalahan dari AWS Elastic Beanstalk Server Elastic Beanstalk mengalami beberapa timeout panggilan API, menyebabkan anomali di simpul hilir.

Aktifkan wawasan di konsol X-Ray

Wawasan harus diaktifkan untuk setiap grup tempat Anda ingin menggunakan fitur wawasan. Anda dapat mengaktifkan wawasan dari halaman Grup.

1. Buka [Konsol X-Ray](#).
2. Pilih grup yang sudah ada atau buat yang baru dengan memilih Buat grup, lalu pilih Aktifkan Wawasan. Untuk informasi selengkapnya tentang mengonfigurasi grup di konsol X-Ray, lihat [Mengkonfigurasi grup](#).
3. Di panel navigasi di sebelah kiri, pilih Wawasan, lalu pilih wawasan untuk ditampilkan.

From

2021-01-18 20:00

To

2021-01-20 11:00

Group

Default

State

All

Apply filter

Description	Duration	Root cause service	Anomalous services	Group	Start time
Overall, 30% of the client requests failed due to faults and 19% of the requests to api (AWS::ElasticBeanstalk::Environment) failed due to faults. Closed Fault	2 minutes 58 seconds	api (AWS::ElasticBeanstalk::Envir...)	www (AWS::ElasticBeanstalk::Envir...) api (AWS::ElasticBeanstalk::Envir...)	Default	Jan 19th 2021, 19:02

Note

X-Ray menggunakan `GetInsightSummaries`, `GetInsight`, `GetInsightEvents`, dan operasi `GetInsightImpactGraph` API untuk mengambil data dari wawasan. Untuk melihat wawasan, gunakan kebijakan terkelola `AWSXray ReadOnlyAccess` IAM atau tambahkan kebijakan kustom berikut ke peran IAM Anda:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:GetInsightSummaries",
        "xray:GetInsight",
        "xray:GetInsightEvents",
        "xray:GetInsightImpactGraph"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Untuk informasi selengkapnya, lihat [Bagaimana AWS X-Ray bekerja dengan IAM](#).

Aktifkan notifikasi wawasan

Dengan notifikasi wawasan, notifikasi dibuat untuk setiap peristiwa wawasan, seperti saat wawasan dibuat, berubah secara signifikan, atau ditutup. Pelanggan dapat menerima pemberitahuan ini melalui EventBridge peristiwa Amazon, dan menggunakan aturan bersyarat untuk mengambil tindakan seperti pemberitahuan SNS, pemanggilan Lambda, memposting pesan ke antrian SQS, atau salah satu target yang mendukung. EventBridge Wawasan notifikasi dipancarkan berdasarkan upaya terbaik - tetapi tidak dijamin. Untuk informasi selengkapnya tentang target, lihat [EventBridge Target Amazon](#).

Anda dapat mengaktifkan notifikasi wawasan untuk setiap grup yang diaktifkan wawasan dari halaman Grup.

Untuk mengaktifkan notifikasi untuk grup X-Ray

1. Buka [Konsol X-Ray](#).
2. Pilih grup yang sudah ada atau buat grup baru dengan memilih Buat grup, pastikan Aktifkan Wawasan dipilih, lalu pilih Aktifkan Pemberitahuan. Untuk informasi selengkapnya tentang mengonfigurasi grup di konsol X-Ray, lihat [Mengkonfigurasi grup](#).

Untuk mengonfigurasi aturan EventBridge bersyarat Amazon

1. Buka [EventBridge konsol Amazon](#).
2. Navigasikan ke Aturan di bilah navigasi sebelah kiri, lalu pilih Buat aturan.
3. Sediakan nama dan deskripsi untuk aturan.
4. Pilih Pola peristiwa, lalu pilih Pola kustom. Sediakan sebuah pola yang berisi "source": ["aws.xray"] dan "detail-type": ["AWS X-Ray Insight Update"]. Berikut ini adalah beberapa contoh pola yang mungkin terjadi.
 - Pola peristiwa untuk mencocokkan semua peristiwa yang masuk dari wawasan X-Ray:

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


- Pola peristiwa untuk mencocokkan **state** dan **category** yang ditentukan:

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

5. Pilih dan konfigurasi target yang ingin Anda panggil ketika suatu peristiwa cocok dengan aturan ini.
6. (Opsional) Berikan tanda untuk mengidentifikasi dan memilih aturan ini dengan lebih mudah.
7. Pilih Buat.

Note

Pemberitahuan wawasan X-Ray mengirimkan peristiwa ke Amazon EventBridge, yang saat ini tidak mendukung kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat [Perlindungan data di AWS X-Ray](#).

Gambaran umum wawasan

Halaman gambaran umum untuk tiga pertanyaan utama yang coba dijawab wawasan:

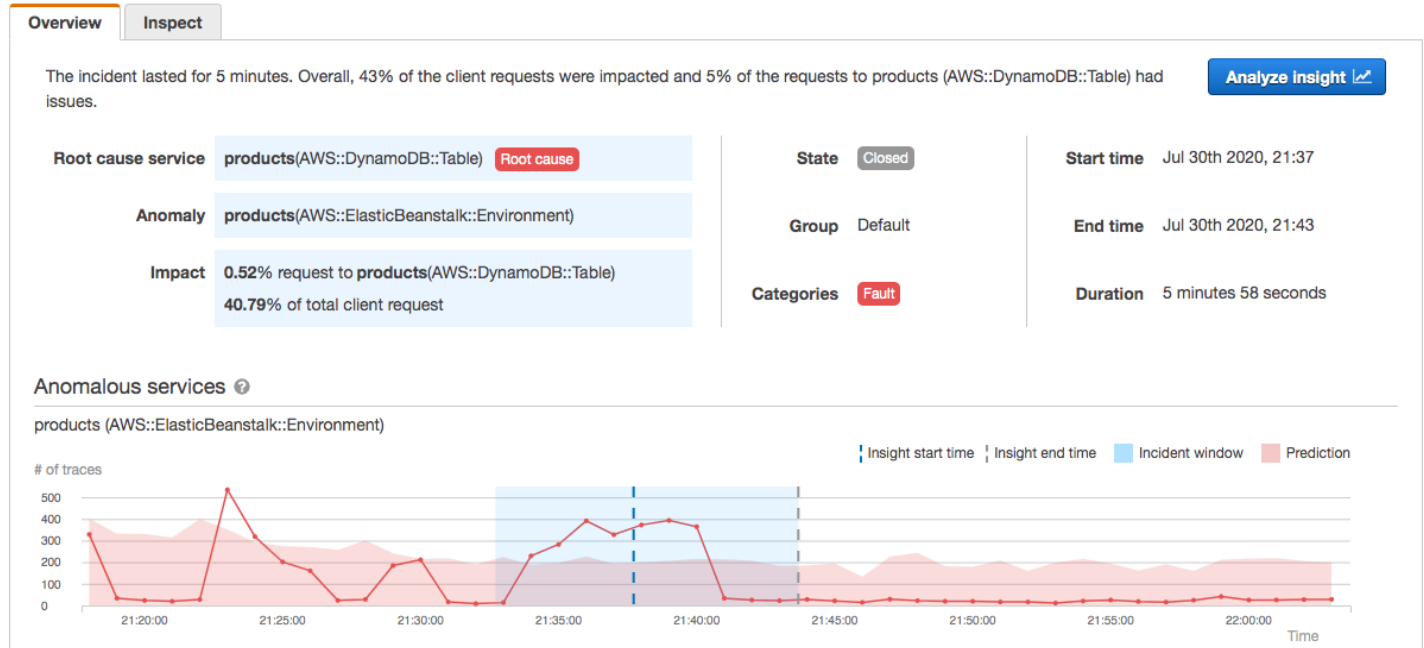
- Apa masalah yang mendasarinya?
- Apa akar penyebabnya?
- Apa dampaknya?

Bagian Layanan anomali menunjukkan garis waktu untuk setiap layanan yang menggambarkan perubahan tingkat kerusakan selama insiden. Garis waktu menunjukkan jumlah jejak dengan kesalahan yang dilapisi pada pita padat yang menunjukkan jumlah kesalahan yang diharapkan berdasarkan jumlah lalu lintas yang direkam. Durasi wawasan divisualisasikan oleh Jendela insiden.

Jendela insiden dimulai saat X-Ray mengamati metrik menjadi anomali dan berlanjut saat wawasan aktif.

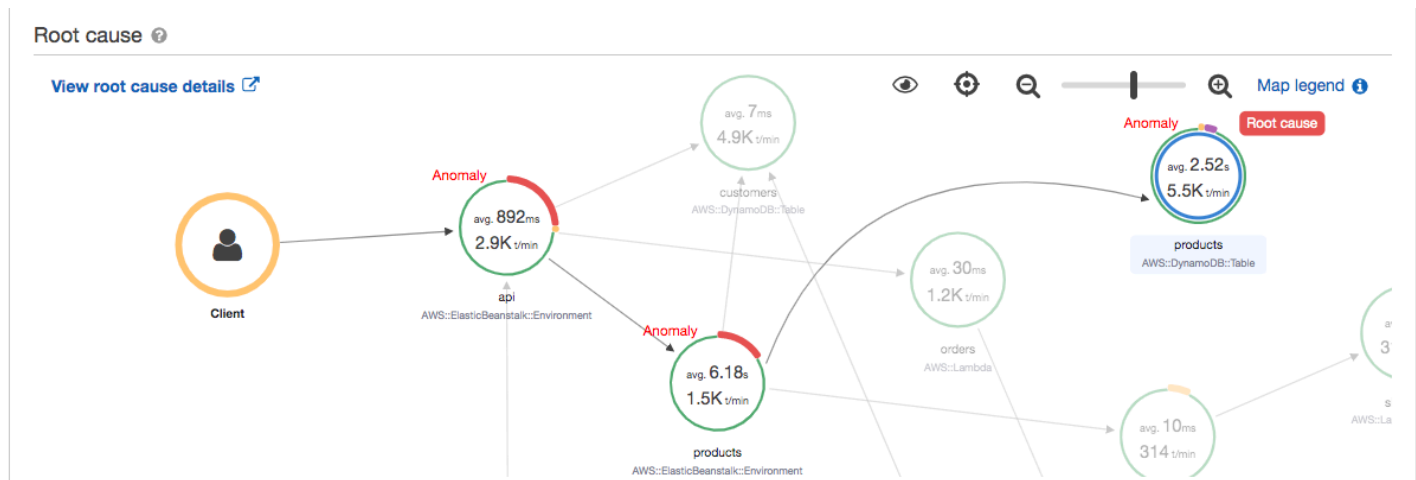
Contoh berikut menunjukkan peningkatan kerusakan yang menyebabkan insiden:

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

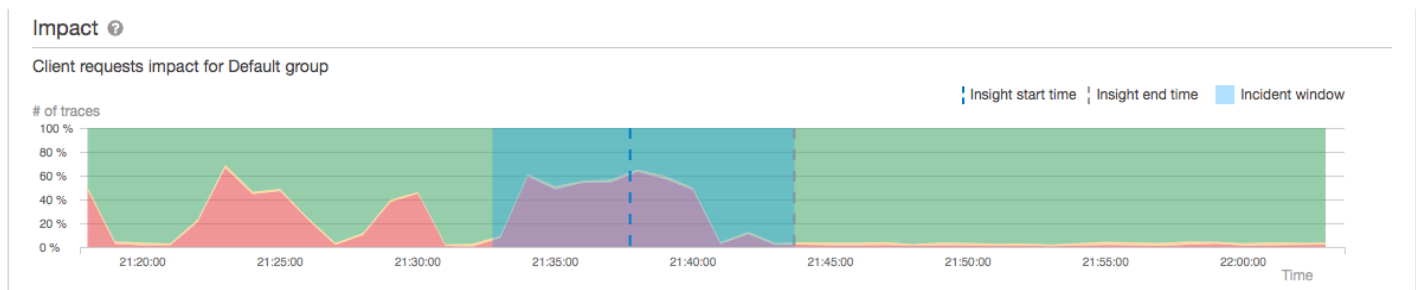


Bagian Root cause menunjukkan peta jejak yang difokuskan pada layanan akar penyebab dan jalur yang terkena dampak. Anda dapat menyembunyikan simpul yang tidak terpengaruh dengan memilih ikon mata di kanan atas peta Akar penyebab. Layanan akar penyebab adalah simpul hilir terjauh tempat X-Ray mengidentifikasi anomali. Layanan ini dapat mewakili layanan yang Anda instrumentasikan atau layanan eksternal yang dipanggil oleh layanan Anda dengan klien yang diinstrumentasikan. Misalnya, jika Anda memanggil Amazon DynamoDB dengan klien SDK yang AWS diinstrumentasi, peningkatan kesalahan dari DynamoDB akan menghasilkan wawasan dengan DynamoDB sebagai akar penyebabnya.

Untuk menyelidiki lebih lanjut akar penyebabnya, pilih Lihat detail akar penyebab pada grafik akar penyebab. Anda dapat menggunakan Analitik untuk menyelidiki akar penyebab dan pesan terkait. Untuk informasi selengkapnya, lihat [Berinteraksi dengan konsol Analitik](#).



Kerusakan yang berlanjut ke hulu di dalam peta dapat memengaruhi beberapa simpul dan menyebabkan beberapa anomali. Jika kerusakan diteruskan sepenuhnya ke pengguna yang membuat permintaan, hasilnya adalah kerusakan client. Ini adalah kesalahan pada simpul akar peta jejak. Grafik Dampak menyediakan lini masa pengalaman klien untuk seluruh grup. Pengalaman ini dihitung berdasarkan persentase status berikut: Kerusakan, Kesalahan, Throttle, dan Oke.



Contoh ini menunjukkan peningkatan pelacakan dengan kerusakan pada simpul akar selama waktu insiden. Insiden di layanan hilir tidak selalu sesuai dengan peningkatan kesalahan klien.

Memilih Analisis wawasan akan membuka konsol Analitik X-Ray di jendela tempat Anda dapat menyelami rangkaian pelacakan yang menyebabkan wawasan. Untuk informasi selengkapnya, lihat [Berinteraksi dengan konsol Analitik](#) .

Memahami dampak

AWS X-Ray mengukur dampak yang disebabkan oleh masalah yang sedang berlangsung sebagai bagian dari menghasilkan wawasan dan pemberitahuan. Dampaknya diukur dengan dua cara:

- Dampak pada kelompok X-Ray
- Dampak pada layanan akar penyebab

Dampak ini ditentukan oleh persentase permintaan yang gagal atau menyebabkan kesalahan dalam jangka waktu tertentu. Analisis dampak ini memungkinkan Anda untuk memperoleh tingkat kepelikan dan prioritas masalah berdasarkan skenario khusus Anda. Dampak ini tersedia sebagai bagian dari pengalaman konsol selain pemberitahuan wawasan.

Deduplikasi

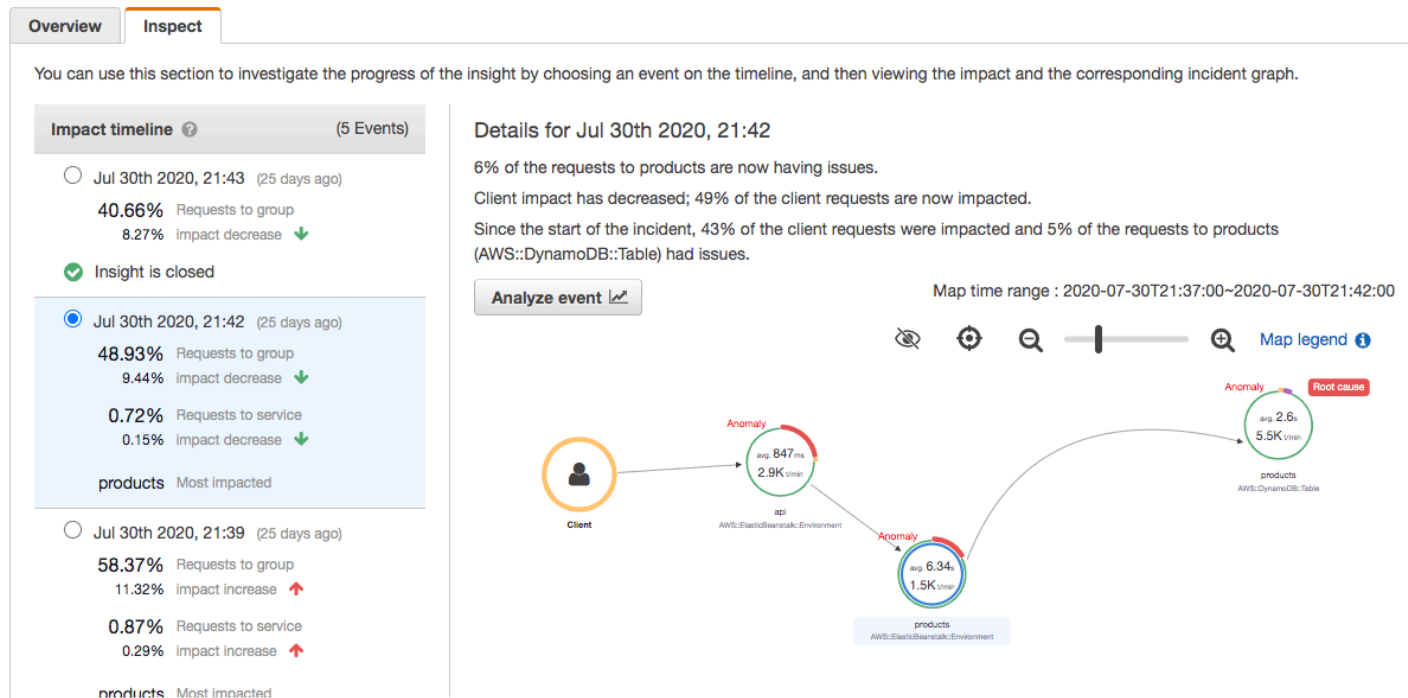
AWS X-Ray wawasan menghilangkan duplikat masalah di beberapa layanan mikro. Ini menggunakan deteksi anomali untuk menentukan layanan yang merupakan akar penyebab masalah, menentukan apakah layanan terkait lainnya menunjukkan perilaku anomali karena akar penyebab yang sama, dan mencatat hasilnya sebagai wawasan tunggal.

Tinjau progres wawasan

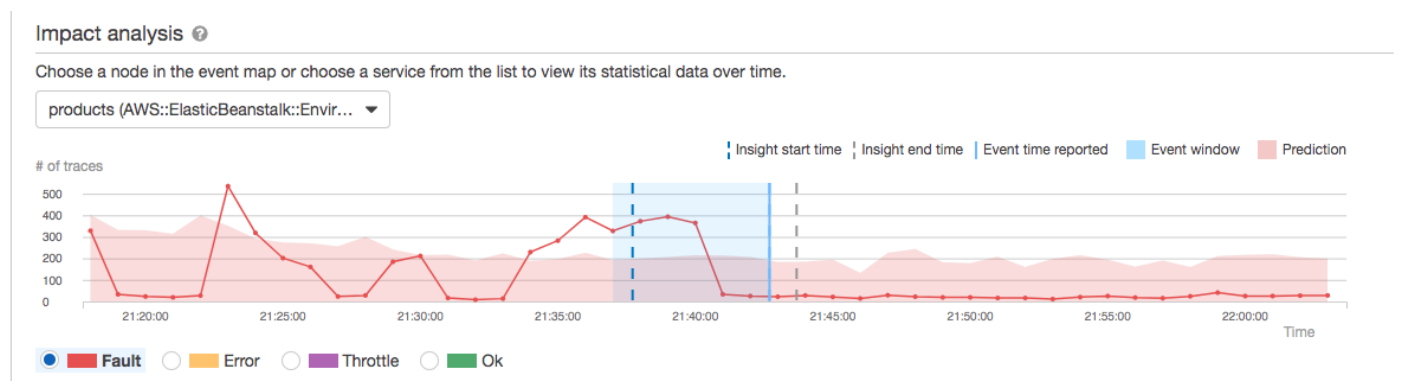
X-Ray mengevaluasi kembali wawasan secara berkala hingga diselesaikan, dan mencatat setiap perubahan perantara yang penting sebagai [pemberitahuan](#), yang dapat dikirim sebagai acara Amazon. EventBridge Hal ini memungkinkan Anda untuk membangun proses dan alur kerja untuk menentukan bagaimana masalah telah berubah dari waktu ke waktu, dan mengambil tindakan yang sesuai seperti mengirim email atau mengintegrasikan dengan sistem peringatan menggunakan. EventBridge

Anda dapat meninjau kejadian peristiwa kejadian di dalam Linimasa Dampak di halaman Inspeksi. Secara default, linimasa menampilkan layanan yang paling terpengaruh hingga Anda memilih layanan yang berbeda.

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



Untuk melihat peta jejak dan grafik untuk suatu peristiwa, pilih dari timeline dampak. Peta jejak menunjukkan layanan dalam aplikasi Anda yang terpengaruh oleh insiden tersebut. Di bawah Analisis dampak, grafik menunjukkan linimasa kerusakan untuk simpul yang dipilih dan untuk klien dalam grup.



Untuk melihat lebih dalam pelacakan yang terlibat dalam insiden, pilih Analisis peristiwa di halaman Inspeksi. Anda dapat menggunakan halaman Analitik untuk memfilter daftar pelacakan dan mengidentifikasi pengguna yang terpengaruh. Untuk informasi selengkapnya, lihat [Berinteraksi dengan konsol Analitik](#).

Berinteraksi dengan konsol Analitik

Konsol AWS X-Ray Analytics adalah alat interaktif untuk menafsirkan data jejak agar cepat memahami kinerja aplikasi Anda dan layanan dasarnya. Konsol ini memungkinkan Anda untuk mengeksplorasi, menganalisis, dan memvisualisasikan pelacakan melalui waktu respons interaktif dan grafik deret waktu.

Saat membuat pilihan di konsol Analitik, konsol tersebut membuat filter untuk mencerminkan subset yang dipilih dari semua pelacakan. Anda dapat mempersempit set data aktif dengan meningkatkan filter terperinci dengan mengeklik grafik dan panel metrik dan bidang yang terkait dengan set pelacakan saat ini.

Topik

- [Fitur konsol](#)
- [Distribusi waktu respons](#)
- [Aktivitas deret waktu](#)
- [Contoh alur kerja](#)
- [Amati kesalahan pada grafik layanan](#)
- [Identifikasi puncak waktu respons](#)
- [Lihat semua pelacakan yang ditandai dengan kode status](#)
- [Lihat semua item dalam subkelompok dan terkait dengan pengguna](#)
- [Bandingkan dua set pelacakan dengan kriteria yang berbeda](#)
- [Identifikasi pelacakan minat dan lihat detailnya](#)

Fitur konsol

Konsol Analitik X-Ray menggunakan fitur kunci berikut untuk mengelompokkan, memfilter, membandingkan, dan mengukur data pelacakan.

Fitur

Fitur	Deskripsi
Grup	Grup yang dipilih di awal adalah Default. Untuk mengubah grup yang diambil, pilih grup yang berbeda dari menu di sebelah kanan

Fitur	Deskripsi
	bilah pencarian ekspresi filter utama. Untuk mempelajari selengkapnya tentang grup lihat, Menggunakan ekspresi filter dengan kelompok .
Jejak yang diambil	Secara default, konsol Analitik menghasilkan grafik berdasarkan semua pelacakan dalam grup yang dipilih. Pelacakan yang diambil mewakili semua pelacakan di set kerja Anda. Anda dapat mencari jumlah pelacakan di file ini. Filter ekspresi yang Anda terapkan ke bilah pencarian utama mempersempit dan memperbarui pelacakan yang diambil.
Tampilkan di bagan/Sembunyikan dari grafik	Beralih untuk membandingkan grup aktif terhadap pelacakan yang diambil. Untuk membandingkan data terkait grup dengan filter aktif apa pun, pilih Tampilkan dalam bagan. Untuk menghilangkan tampilan ini dari bagan, pilih Sembunyikan dari bagan.
Jejak yang difilter set A	Melalui interaksi dengan grafik dan tabel, terapkan filter untuk membuat kriteria untuk Set pelacakan yang difilter. Ketika filter diterapkan, jumlah pelacakan yang berlaku dan persentase pelacakan dari total yang diambil dihitung dalam file ini. Filter diisi sebagai tanda dalam ubin Set pelacakan A yang difilter dan juga dapat dihapus dari ubin.

Fitur	Deskripsi
Perbaiki	Fungsi ini memperbarui set pelacakan yang diambil berdasarkan filter yang diterapkan untuk melacak set A. Penyempitan set pelacakan yang diambil menyegarkan set pekerjaan dari semua pelacakan yang diambil berdasarkan filter untuk set pelacakan A. set kerja pelacakan yang diambil adalah subset sampel dari semua pelacakan dalam grup.
Set jejak yang difilter B	Saat dibuat, kumpulan jejak yang difilter B adalah salinan dari Jejak yang difilter set A. Untuk membandingkan dua set jejak, buat pilihan filter baru yang akan berlaku untuk melacak set B, sementara trace set A tetap diperbaiki. Ketika filter diterapkan, jumlah pelacakan yang berlaku dan persentase pelacakan dari total yang diambil dihitung dalam ubin ini. Filter diisi sebagai tanda dalam ubin Set pelacakan B yang difilter dan juga dapat dihapus dari ubin.
Jalur entitas akar penyebab waktu respons	Sebuah tabel jalur entitas yang dicatat. X-Ray menentukan jalur di pelacakan Anda yang merupakan penyebab yang paling memungkinkan untuk waktu respons. Format menunjukkan hierarki entitas yang ditemui, berakhir dengan akar masalah waktu respons. Gunakan baris ini untuk memfilter kesalahan waktu respons berulang. Untuk informasi selengkapnya tentang menyesuaikan filter akar masalah dan mendapatkan data melalui API lihat, Mengambil dan menyempitkan analitik akar masalah .

Fitur	Deskripsi
Delta ()	Kolom yang ditambahkan ke tabel metrik ketika kedua set pelacakan A dan set pelacakan B aktif. Kolom Delta menghitung selisih dalam persentase pelacakan antara pelacakan antara set pelacakan A dan set pelacakan B.

Distribusi waktu respons

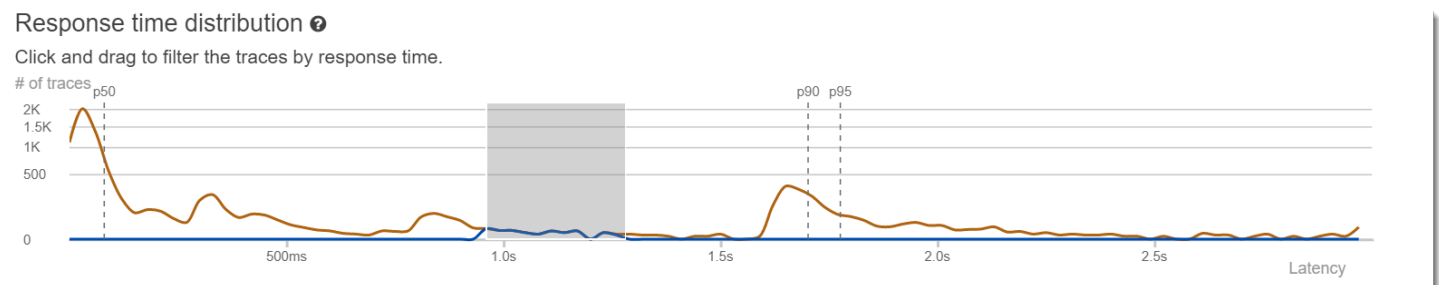
Konsol Analitik X-Ray menghasilkan dua grafik utama untuk membantu Anda memvisualisasikan pelacakan: Distribusi Waktu Respons dan Aktivitas Deret Waktu. Bagian ini dan berikut ini memberikan contoh masing-masing, dan menjelaskan dasar-dasar cara membaca grafik.

Berikut ini adalah warna yang terkait dengan grafik garis waktu respons (grafik deret waktu menggunakan skema warna yang sama):

- Semua pelacakan dalam grup – abu-abu
- Pelacakan yang diambil – oranye
- Set pelacakan A yang difilter – hijau
- Set pelacakan B yang difilter – biru

Example – Distribusi waktu respons

Distribusi waktu respons adalah bagan yang menunjukkan jumlah pelacakan dengan waktu respons yang diberikan. Klik dan seret untuk membuat pilihan dalam distribusi waktu respons. Hal ini memilih dan membuat filter pada set pelacakan kerja bernama `responseTime` untuk semua pelacakan dalam waktu respons tertentu.

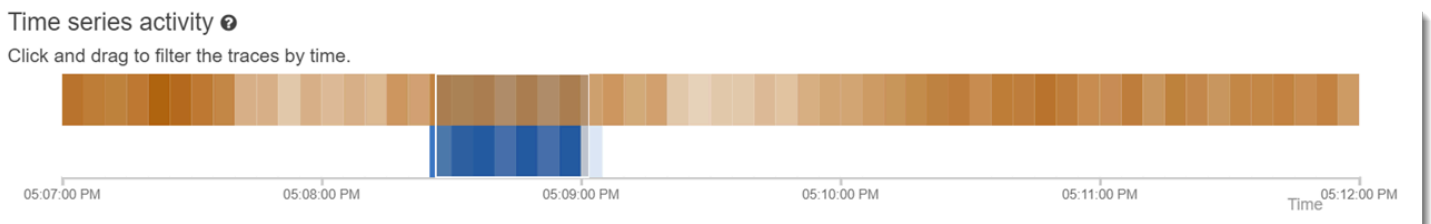


Aktivitas deret waktu

Bagan aktivitas deret waktu menunjukkan jumlah pelacakan pada periode waktu tertentu. Indikator warna mencerminkan warna grafik garis dari distribusi waktu respons. Semakin gelap dan penuh blok warna dalam rangkaian aktivitas, semakin banyak pelacakan yang ditunjukkan pada waktu tertentu.

Example – Aktivitas deret waktu

Klik dan seret untuk membuat pilihan dalam grafik aktivitas deret waktu. Hal ini memilih dan membuat filter bernama `timerange` pada set pelacakan kerja untuk semua pelacakan dalam kisaran waktu tertentu.



Contoh alur kerja

Contoh berikut menunjukkan kasus penggunaan umum untuk konsol Analitik X-Ray. Setiap contoh menunjukkan fungsi kunci dari pengalaman konsol tersebut. Sebagai grup, contoh mengikuti alur kerja pemecahan masalah dasar. Langkah-langkah tersebut menjelaskan cara pertama menemukan simpul yang tidak sehat, lalu cara berinteraksi dengan konsol Analitik untuk secara otomatis menghasilkan kueri komparatif. Setelah Anda mempersempit ruang lingkup melalui kueri, Anda akhirnya akan melihat detail pelacakan yang menarik untuk menentukan apa yang kondisi layanan Anda.

Amati kesalahan pada grafik layanan

Peta jejak menunjukkan kesehatan setiap node dengan mewarnai berdasarkan rasio panggilan yang berhasil terhadap kesalahan dan kesalahan. Ketika persentase merah terlihat pada simpul Anda, warna tersebut menandakan adanya kerusakan. Gunakan konsol Analitik X-Ray untuk menyelidikinya.

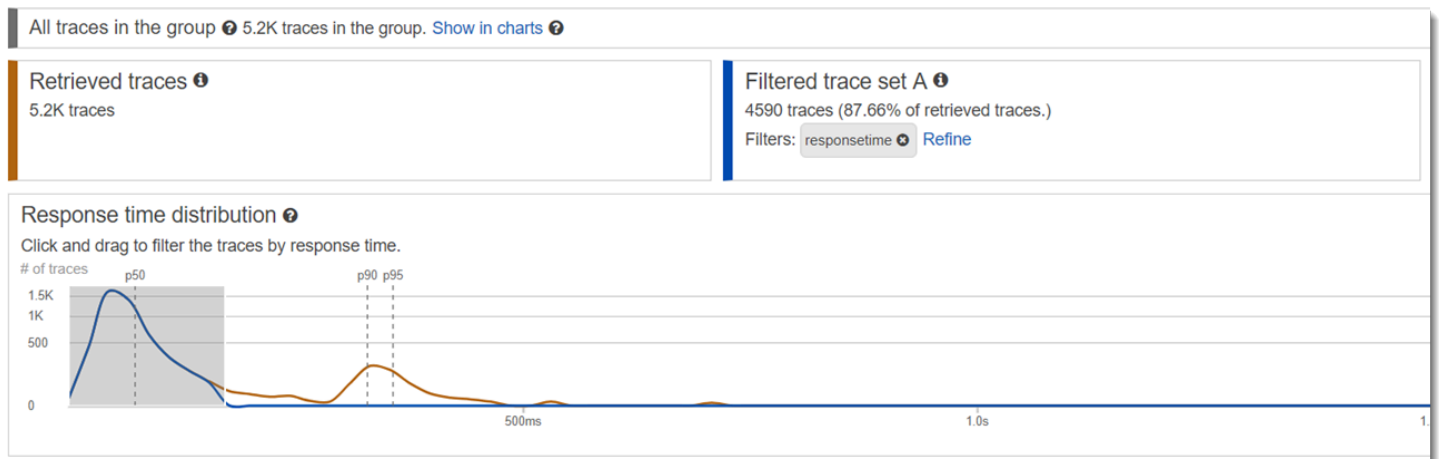
Untuk informasi selengkapnya tentang cara membaca peta jejak, lihat [Melihat peta jejak](#).



Identifikasi puncak waktu respons

Menggunakan distribusi waktu respons, Anda dapat mengamati puncak dalam waktu respons. Dengan memilih puncak dalam waktu respons, tabel di bawah grafik akan diperbarui untuk mengekspos semua metrik terkait, seperti kode status.

Saat Anda mengklik dan menyeret, X-Ray akan memilih dan membuat filter. Ini ditampilkan dalam bayangan abu-abu di atas garis grafik. Sekarang Anda dapat menyeret bayangan tersebut ke kiri dan kanan sepanjang distribusi untuk memperbarui pilihan dan filter Anda.



Lihat semua pelacakan yang ditandai dengan kode status

Anda dapat menelusuri pelacakan dalam puncak yang dipilih menggunakan tabel metrik di bawah grafik. Dengan mengklik baris di tabel KODE STATUS HTTP, Anda secara otomatis membuat filter pada set data bekerja. Misalnya, Anda dapat melihat semua pelacakan kode status 500. Hal ini menciptakan tanda filter di ubin set pelacakan bernama `http.status`.

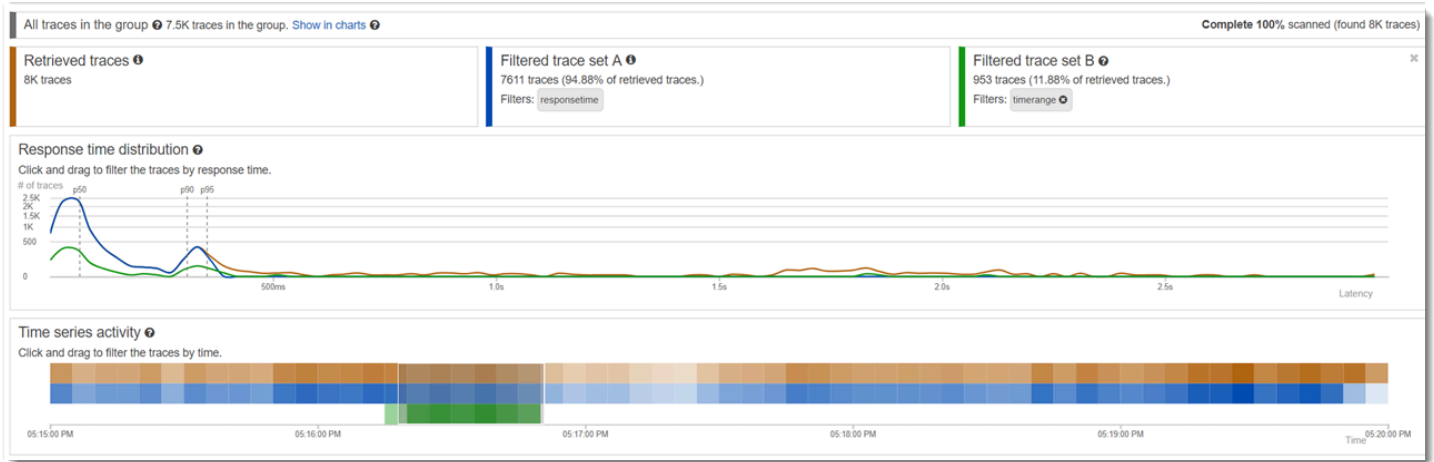
Lihat semua item dalam subkelompok dan terkait dengan pengguna

Telusuri set kesalahan berdasarkan pengguna, URL, akar masalah waktu respons, atau atribut lain yang ditentukan sebelumnya. Misalnya, untuk memfilter set pelacakan dengan kode status 500, pilih baris dari tabel PENGGUNA. Hal ini menyebabkan dua tanda filter di ubin set pelacakan: `http.status`, seperti yang ditetapkan sebelumnya, dan `user`.

Bandingkan dua set pelacakan dengan kriteria yang berbeda

Bandingkan berbagai pengguna dengan permintaan POST mereka untuk menemukan perbedaan dan korelasi lainnya. Terapkan set filter pertama Anda. Set filter ditentukan berdasarkan garis biru dalam distribusi waktu respons. Kemudian pilih Bandingkan. Awalnya, ini hal ini membuat salinan filter pada set pelacakan set A.

Untuk melanjutkan, tentukan satu set filter baru untuk diterapkan pada set pelacakan B. set kedua ini diwakili oleh garis hijau. Contoh berikut menunjukkan garis yang berbeda berdasarkan skema warna biru dan hijau.



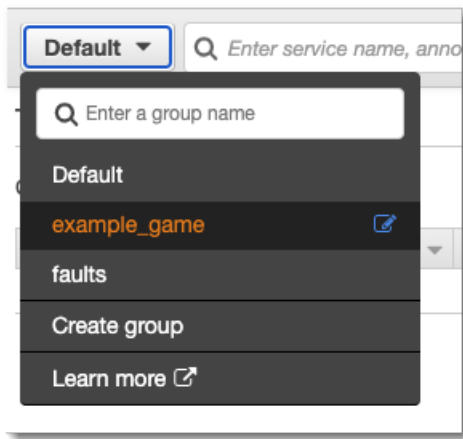
Identifikasi pelacakan minat dan lihat detailnya

Saat Anda mempersempit cakupan Anda menggunakan filter konsol tersebut, daftar pelacakan di bawah tabel metrik menjadi lebih bermakna. Tabel daftar pelacakan menggabungkan informasi tentang URL, PENGGUNA, dan KODE STATUS menjadi satu tampilan. Untuk wawasan lebih lanjut, pilih baris dari tabel ini untuk membuka halaman detail pelacakan dan lihat lini masa dan data mentah.

Mengkonfigurasi grup

Grup adalah kumpulan pelacakan yang ditentukan oleh ekspresi filter. Anda dapat menggunakan grup untuk menghasilkan grafik layanan tambahan dan menyediakan CloudWatch metrik Amazon. Anda dapat menggunakan konsol AWS X-Ray tersebut atau API X-Ray untuk membuat dan mengelola grup untuk layanan Anda. Topik ini menjelaskan cara membuat dan mengelola grup menggunakan konsol X-Ray. Untuk informasi tentang cara mengelola grup dengan menggunakan API X-Ray, lihat [Grup](#).

Anda dapat membuat grup jejak untuk peta jejak, jejak, atau analitik. Saat Anda membuat grup, grup akan tersedia sebagai filter pada menu tarik-turun grup di ketiga halaman: Trace Map, Traces, dan Analytics.



Grup diidentifikasi oleh nama mereka atau Amazon Resource Name (ARN), dan berisi ekspresi filter. Layanan ini membandingkan pelacakan masuk ke ekspresi dan menyimpannya dengan sesuai. Untuk informasi selengkapnya tentang cara membangun ekspresi filter, lihat [Menggunakan ekspresi filter](#).

Memperbarui ekspresi filter grup tidak mengubah data yang sudah dicatat. Pembaruan hanya berlaku untuk pelacakan berikutnya. Hal ini dapat mengakibatkan grafik gabungan dari ekspresi baru dan lama. Untuk menghindari hal tersebut, hapus grup saat ini dan buat yang baru.

Note

Grup ditagih berdasarkan jumlah pelacakan yang diambil yang cocok dengan ekspresi filter. Untuk informasi selengkapnya, lihat [Harga AWS X-Ray](#).

Topik

- [Membuat grup](#)
- [Terapkan grup](#)
- [Mengedit grup](#)
- [Klon grup](#)
- [Menghapus grup](#)
- [Lihat metrik grup di Amazon CloudWatch](#)

Membuat grup

Note

Anda sekarang dapat mengonfigurasi grup X-Ray dari dalam CloudWatch konsol Amazon. Anda juga dapat terus menggunakan konsol X-Ray.

CloudWatch console

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Pengaturan di panel navigasi kiri.
3. Pilih Lihat pengaturan di bawah Grup dalam bagian jejak X-Ray.
4. Pilih Buat grup di atas daftar grup.
5. Pada halaman Buat grup, masukkan nama untuk grup. Nama grup dapat memiliki maksimum 32 karakter, dan berisi karakter alfanumerik serta tanda hubung. Nama grup peka terhadap huruf besar atau kecil.
6. Memasukkan ekspresi filter. Untuk informasi selengkapnya tentang cara membangun ekspresi filter, lihat [Menggunakan ekspresi filter](#). Pada contoh berikut, grup filter untuk pelacakan kesalahan dari layanan `api.example.com` dan permintaan ke layanan saat waktu respons lebih besar dari atau sama dengan lima detik.

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

7. Pada Wawasan, aktifkan atau nonaktifkan akses wawasan untuk grup. Untuk informasi selengkapnya tentang wawasan, lihat [Menggunakan wawasan X-Ray](#).

☒ Enable insights

☐ Enable notifications

Deliver insight events using Amazon EventBridge.

8. Di Tag, pilih Tambahkan tag baru untuk memasukkan kunci tag, dan secara opsional, nilai tag. Lanjutkan untuk menambahkan tag tambahan seperti yang diinginkan. Kunci tanda harus unik. Untuk menghapus tag, pilih Hapus di bawah setiap tag. Untuk informasi selengkapnya tentang tanda, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

Key	Value - optional
Q Enter key	Q Enter value
Remove	

9. Pilih Buat grup.

X-Ray console

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Buka halaman Buat grup dari halaman Grup di panel navigasi kiri, atau dari menu grup di salah satu halaman berikut: Lacak Peta, Jejak, dan Analisis.
3. Pada halaman Buat grup, masukkan nama untuk grup. Nama grup dapat memiliki maksimum 32 karakter, dan berisi karakter alfanumerik serta tanda hubung. Nama grup peka terhadap huruf besar atau kecil.
4. Memasukkan ekspresi filter. Untuk informasi selengkapnya tentang cara membangun ekspresi filter, lihat [Menggunakan ekspresi filter](#). Pada contoh berikut, grup filter untuk pelacakan kesalahan dari layananapi.example.com. dan permintaan ke layanan saat waktu respons lebih besar dari atau sama dengan lima detik.

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

5. Pada Wawasan, aktifkan atau nonaktifkan akses wawasan untuk grup. Untuk informasi selengkapnya tentang wawasan, lihat [Menggunakan wawasan X-Ray](#).

Enable Insights ☒

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

6. Di Tanda, masukkan kunci tanda dan, secara opsional, nilai tanda. Saat Anda menambahkan tanda, baris baru akan muncul untuk memasukkan tanda lain. Kunci tanda harus unik. Untuk menghapus tanda, pilih X di akhir baris tanda. Untuk informasi selengkapnya tentang tanda, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

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

7. Pilih Buat grup.

Terapkan grup

CloudWatch console

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Buka salah satu halaman berikut dari panel navigasi di bawah jejak X-Ray:
 - Peta Jejak
 - Jejak
3. Masukkan nama grup ke dalam filter grup Filter by X-Ray. Data yang ditampilkan pada halaman berubah agar sesuai dengan ekspresi filter yang ditetapkan dalam grup.

X-Ray console

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Buka salah satu halaman berikut dari panel navigasi :
 - Peta Jejak
 - Jejak
 - Analitik
3. Pada menu grup, pilih grup yang Anda buat di [the section called “Membuat grup”](#). Data yang ditampilkan pada halaman berubah agar sesuai dengan ekspresi filter yang ditetapkan dalam grup.

Mengedit grup

CloudWatch console

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Pengaturan di panel navigasi kiri.
3. Pilih Lihat pengaturan di bawah Grup dalam bagian jejak X-Ray.
4. Pilih grup dari bagian Grup dan kemudian pilih Edit.
5. Meskipun Anda tidak dapat mengubah nama grup, Anda dapat memperbarui ekspresi filter. Untuk informasi selengkapnya tentang cara membangun ekspresi filter, lihat [Menggunakan ekspresi filter](#). Dalam contoh berikut, grup memfilter pelacakan kesalahan dari layanan `api.example.com`, yang berisi alamat URL permintaan `example/game`, dan waktu respons untuk permintaan lebih besar dari atau sama dengan lima detik.

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

6. Pada Wawasan, aktifkan atau nonaktifkan akses wawasan untuk grup. Untuk informasi selengkapnya tentang wawasan, lihat [Menggunakan wawasan X-Ray](#).

☒ Enable insights

☐ Enable notifications

Deliver insight events using Amazon EventBridge.

7. Di Tag, pilih Tambahkan tag baru untuk memasukkan kunci tag, dan secara opsional, nilai tag. Lanjutkan untuk menambahkan tag tambahan seperti yang diinginkan. Kunci tanda harus unik. Untuk menghapus tag, pilih Hapus di bawah setiap tag. Untuk informasi selengkapnya tentang tanda, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

Key	Value - optional
Q Enter key	Q Enter value
Remove	

8. Setelah selesai memperbarui grup, pilih Perbarui grup.

X-Ray console

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Lakukan salah satu langkah berikut untuk membuka halaman Mengedit grup.
 - a. Pada halaman Grup, pilih nama grup untuk mengeditnya.
 - b. Pada menu grup di salah satu halaman berikut, arahkan ke grup, lalu pilih Edit.
 - Peta Jejak
 - Jejak
 - Analitik
3. Meskipun Anda tidak dapat mengubah nama grup, Anda dapat memperbarui ekspresi filter. Untuk informasi selengkapnya tentang cara membangun ekspresi filter, lihat [Menggunakan ekspresi filter](#). Dalam contoh berikut, grup memfilter pelacakan kesalahan dari layanan `api.example.com`, yang berisi alamat URL permintaan `example/game`, dan waktu respons untuk permintaan lebih besar dari atau sama dengan lima detik.

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

4. Di Wawasan, aktifkan atau nonaktifkan wawasan dan notifikasi wawasan untuk grup. Untuk informasi selengkapnya tentang wawasan, lihat [Menggunakan wawasan X-Ray](#).

Enable Insights ☒

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

5. Di Tanda, edit kunci dan nilai tanda. Kunci tanda harus unik. Nilai tanda adalah opsional; Anda dapat menghapus nilai-nilai, jika Anda ingin. Untuk menghapus tanda, pilih X di akhir baris tanda. Untuk informasi selengkapnya tentang tanda, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

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

6. Setelah selesai memperbarui grup, pilih Perbarui grup.

Klon grup

Meng-klon grup membuat grup baru yang memiliki ekspresi filter dan tanda dari grup yang ada. Ketika Anda meng-klon grup, grup baru yang memiliki nama yang sama dengan grup yang di-klon, dengan `-clone` ditambahkan ke nama.

CloudWatch console

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Pengaturan di panel navigasi kiri.
3. Pilih Lihat pengaturan di bawah Grup dalam bagian jejak X-Ray.
4. Pilih grup dari bagian Grup dan kemudian pilih Klon.
5. Pada halaman Buat grup, nama grup adalah `group-name-clone`. Secara opsional, masukkan nama baru untuk grup. Nama grup dapat memiliki maksimum 32 karakter, dan berisi karakter alfanumerik serta tanda hubung. Nama grup peka terhadap huruf besar atau kecil.
6. Anda dapat menjaga ekspresi filter dari grup yang ada, atau secara opsional, memasukkan ekspresi filter baru. Untuk informasi selengkapnya tentang cara membangun ekspresi filter, lihat [Menggunakan ekspresi filter](#). Pada contoh berikut, grup filter untuk pelacakan kesalahan dari `layanapi.example.com` dan permintaan ke layanan saat waktu respons lebih besar dari atau sama dengan lima detik.

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

7. Di Tanda, edit kunci tanda dan nilai, jika diperlukan. Kunci tanda harus unik. Nilai tanda adalah opsional; Anda dapat menghapus nilai jika Anda ingin. Untuk menghapus tanda, pilih X di akhir baris tanda. Untuk informasi selengkapnya tentang tanda, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).
8. Pilih Buat grup.

X-Ray console

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Buka halaman Grup dari panel navigasi kiri, dan pilih nama grup yang ingin Anda kloning.

3. Pilih Clone group dari menu Actions.
4. Pada halaman Buat grup, nama grup adalah `group-name-clone`. Secara opsional, masukkan nama baru untuk grup. Nama grup dapat memiliki maksimum 32 karakter, dan berisi karakter alfanumerik serta tanda hubung. Nama grup peka terhadap huruf besar atau kecil.
5. Anda dapat menjaga ekspresi filter dari grup yang ada, atau secara opsional, memasukkan ekspresi filter baru. Untuk informasi selengkapnya tentang cara membangun ekspresi filter, lihat [Menggunakan ekspresi filter](#). Pada contoh berikut, grup filter untuk pelacakan kesalahan dari `layanapi.example.com` dan permintaan ke layanan saat waktu respons lebih besar dari atau sama dengan lima detik.

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

6. Di Tanda, edit kunci tanda dan nilai, jika diperlukan. Kunci tanda harus unik. Nilai tanda adalah opsional; Anda dapat menghapus nilai jika Anda ingin. Untuk menghapus tanda, pilih X di akhir baris tanda. Untuk informasi selengkapnya tentang tanda, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).
7. Pilih Buat grup.

Menghapus grup

Ikuti langkah-langkah di bagian ini untuk menghapus grup. Anda tidak dapat menghapus grup Default.

CloudWatch console

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Pengaturan di panel navigasi kiri.
3. Pilih Lihat pengaturan di bawah Grup dalam bagian jejak X-Ray.
4. Pilih grup dari bagian Grup dan kemudian pilih Hapus.
5. Ketika Anda diminta untuk mengonfirmasi, pilih Hapus.

X-Ray console

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Buka halaman Grup dari panel navigasi kiri, dan pilih nama grup yang ingin Anda hapus.
3. Dari menu Tindakan, pilih Hapus grup.
4. Ketika Anda diminta untuk mengonfirmasi, pilih Hapus.

Lihat metrik grup di Amazon CloudWatch

Setelah grup dibuat, pelacakan masuk diperiksa terhadap ekspresi filter grup saat disimpan di layanan X-Ray. Metrik untuk jumlah jejak yang cocok dengan setiap kriteria dipublikasikan ke Amazon CloudWatch setiap menit. Memilih Lihat metrik pada halaman Edit grup membuka CloudWatch konsol ke halaman Metrik. Untuk informasi selengkapnya tentang cara menggunakan CloudWatch metrik, lihat [Menggunakan CloudWatch Metrik Amazon](#) di CloudWatch Panduan Pengguna Amazon.

CloudWatch console

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Pengaturan di panel navigasi kiri.
3. Pilih Lihat pengaturan di bawah Grup dalam bagian jejak X-Ray.
4. Pilih grup dari bagian Grup dan kemudian pilih Edit.
5. Pada halaman Edit grup, pilih Lihat metrik.

Halaman Metrik CloudWatch konsol terbuka di tab baru.

X-Ray console

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Buka halaman Grup dari panel navigasi kiri, dan pilih nama grup yang ingin Anda lihat metriknya.
3. Pada halaman Edit grup, pilih Lihat metrik.

Halaman Metrik CloudWatch konsol terbuka di tab baru.

Mengonfigurasi aturan pengambilan sampel

Anda dapat menggunakan AWS X-Ray konsol untuk mengonfigurasi aturan pengambilan sampel untuk layanan Anda. X-Ray SDK dan Layanan AWS yang mendukung [penelusuran aktif](#) dengan konfigurasi sampling menggunakan aturan pengambilan sampel untuk menentukan permintaan mana yang akan direkam.

Topik

- [Mengonfigurasi aturan pengambilan sampel](#)
- [Menyesuaikan aturan pengambilan sampel](#)
- [Opsi aturan pengambilan sampel](#)
- [Contoh aturan pengambilan sampel](#)
- [Mengonfigurasi layanan Anda untuk menggunakan aturan pengambilan sampel](#)
- [Melihat hasil pengambilan sampel](#)
- [Langkah selanjutnya](#)

Mengonfigurasi aturan pengambilan sampel

Anda dapat mengonfigurasi pengambilan sampel untuk kasus penggunaan berikut ini:

- Titik Masuk API Gateway – API Gateway mendukung pengambilan sampel dan pelacakan aktif. Untuk mengaktifkan pelacakan aktif pada tahap API, lihat [Dukungan penelusuran aktif Amazon API Gateway untuk AWS X-Ray](#).
- AWS AppSync— AWS AppSync mendukung pengambilan sampel dan penelusuran aktif. Untuk mengaktifkan penelusuran aktif pada AWS AppSync permintaan, lihat [Menelusuri dengan AWS X-Ray](#).
- Instrumen X-Ray SDK pada platform komputasi — Saat menggunakan platform komputasi seperti Amazon, EC2 Amazon ECS, atau AWS Elastic Beanstalk, pengambilan sampel didukung saat aplikasi telah diinstrumentasi dengan SDK X-Ray terbaru.

Menyesuaikan aturan pengambilan sampel

Dengan menyesuaikan aturan pengambilan sampel, Anda dapat mengontrol jumlah data yang Anda rekam. Anda juga dapat memodifikasi perilaku pengambilan sampel tanpa memodifikasi atau menerapkan ulang kode Anda. Aturan pengambilan sampel memberi tahu SDK X-Ray jumlah permintaan yang harus dicatat untuk satu set kriteria. Secara default, SDK X-Ray mencatat permintaan pertama setiap detik, dan lima persen dari setiap permintaan tambahan. Satu permintaan per detik adalah reservoir. Tindakan ini memastikan bahwa setidaknya satu pelacakan dicatat setiap detik selama layanan melayani permintaan. Lima persen adalah tingkat saat permintaan tambahan di luar ukuran reservoir diambil sampelnya.

Anda dapat mengonfigurasi SDK X-Ray untuk membaca aturan pengambilan sampel dari dokumen JSON yang Anda sertakan dengan kode Anda. Namun, saat Anda menjalankan beberapa instans layanan Anda, setiap instans melakukan pengambilan sampel secara independen. Hal ini menyebabkan persentase keseluruhan permintaan sampel meningkat karena reservoir dari semua instans secara efektif ditambahkan bersama-sama. Selain itu, untuk memperbarui aturan pengambilan sampel lokal, Anda harus menerapkan ulang kode Anda.

Dengan menetapkan aturan pengambilan sampel di konsol X-Ray, dan [mengonfigurasi SDK](#) untuk membaca aturan dari layanan X-Ray, Anda dapat menghindari kedua masalah ini. Layanan mengelola reservoir untuk setiap aturan, dan menetapkan kuota ke setiap instans layanan Anda untuk mendistribusikan reservoir secara merata, berdasarkan jumlah instans yang berjalan. Batas reservoir dihitung sesuai dengan aturan yang Anda tetapkan. Karena aturan dikonfigurasi dalam layanan, Anda dapat mengelola aturan tanpa membuat penerapan tambahan.

Note

X-Ray menggunakan pendekatan upaya terbaik dalam menerapkan aturan pengambilan sampel, dan dalam beberapa kasus laju pengambilan sampel efektif mungkin tidak sama persis dengan aturan pengambilan sampel yang dikonfigurasi. Namun, seiring waktu jumlah permintaan sampel harus mendekati persentase yang dikonfigurasi.

Anda sekarang dapat mengonfigurasi aturan pengambilan sampel X-Ray dari dalam CloudWatch konsol Amazon. Anda juga dapat terus menggunakan konsol X-Ray.

CloudWatch console

Untuk mengonfigurasi aturan pengambilan sampel di konsol CloudWatch

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Pengaturan di panel navigasi kiri.
3. Pilih Lihat pengaturan di bawah Aturan pengambilan sampel dalam bagian jejak X-Ray.
4. Untuk membuat aturan, pilih Buat aturan pengambilan sampel.

Untuk mengedit aturan, pilih aturan dan pilih Edit untuk mengeditnya.

Untuk menghapus aturan, pilih aturan dan pilih Hapus untuk menghapusnya.

X-Ray console

Untuk mengonfigurasi aturan pengambilan sampel di konsol X-Ray

1. Buka [Konsol X-Ray](#).
2. Pilih Sampling di panel navigasi kiri.
3. Untuk membuat aturan, pilih Buat aturan pengambilan sampel.

Untuk mengedit aturan, pilih nama aturan.

Untuk menghapus aturan, pilih aturan dan gunakan menu Tindakan untuk menghapusnya.

Opsi aturan pengambilan sampel

Opsi berikut tersedia untuk setiap aturan. Nilai string dapat menggunakan wildcard untuk mencocokkan satu karakter (?) atau nol atau beberapa karakter (*).

Opsi aturan pengambilan sampel

- Nama aturan (string) - Sebuah nama unik untuk aturan tersebut.
- Prioritas (bilangan bulat antara 1 hingga 9999) - Prioritas aturan pengambilan sampel. Layanan mengevaluasi aturan dalam urutan prioritas naik, dan membuat keputusan pengambilan sampel dengan aturan pertama yang cocok.

- Reservoir (bilangan bulat bukan negatif) – Jumlah tetap permintaan yang cocok dengan instrumen per detik, sebelum menerapkan tingkat tetap. Reservoir tidak digunakan secara langsung oleh layanan, tetapi berlaku untuk semua layanan yang menggunakan aturan secara kolektif.
- Tingkat (bilangan bulat antara 0 dan 100) — Persentase permintaan yang cocok dengan instrumen, setelah reservoir habis. Saat mengonfigurasi aturan pengambilan sampel di konsol, pilih persentase antara 0 dan 100. Saat mengonfigurasi aturan pengambilan sampel dalam SDK klien menggunakan dokumen JSON, berikan nilai persentase antara 0 dan 1.
- Nama layanan (string) — Nama layanan instrumentasi, seperti yang muncul di peta jejak.
 - SDK X-Ray – Nama layanan yang Anda konfigurasikan pada pencatat.
 - Amazon API Gateway – *api-name/stage*.
- Jenis layanan (string) - Jenis layanan, seperti yang muncul di peta jejak. Untuk SDK X-Ray, atur tipe layanan dengan menerapkan plugin yang sesuai:
 - `AWS::ElasticBeanstalk::Environment`— AWS Elastic Beanstalk Lingkungan (plugin).
 - `AWS::EC2::Instance`— Sebuah EC2 contoh Amazon (plugin).
 - `AWS::ECS::Container` – Sebuah kontainer Amazon ECS (plugin).
 - `AWS::APIGateway::Stage` – Sebuah tahap Amazon API Gateway.
 - `AWS::AppSync::GraphQLAPI` — Permintaan AWS AppSync API.
- Host (string) – Nama host dari header host HTTP.
- Metode HTTP (string) – Metode permintaan HTTP.
- Jalur URL (string) – Jalur URL permintaan.
 - SDK X-Ray – Bagian jalur dari URL permintaan HTTP.
- Resource ARN (string) — ARN AWS sumber daya yang menjalankan layanan.
 - SDK X-Ray – Tidak didukung. SDK hanya dapat menggunakan aturan dengan ARN Sumber Daya diatur ke `*`.
 - Amazon API Gateway – ARN tahap.
- (Opsional) Atribut (kunci dan nilai) – Atribut segmen yang diketahui ketika keputusan pengambilan sampel dibuat.
 - SDK X-Ray – Tidak didukung. SDK mengabaikan aturan yang menentukan atribut.
 - Amazon API Gateway – Header dari permintaan HTTP asli.

Contoh aturan pengambilan sampel

Example – Aturan default tanpa reservoir dan tingkat rendah

Anda dapat mengubah reservoir dan tingkat aturan default. Aturan default berlaku untuk permintaan yang tidak cocok dengan aturan lainnya.

- Waduk: **0**
- Nilai: **5** (**0.05** jika dikonfigurasi menggunakan dokumen JSON)

Example – Aturan men-debug untuk melacak semua permintaan untuk rute yang bermasalah

Aturan prioritas tinggi diterapkan sementara untuk men-debug.

- Nama aturan: **DEBUG – history updates**
- Prioritas: **1**
- Waduk: **1**
- Nilai: **100** (**1** jika dikonfigurasi menggunakan dokumen JSON)
- Nama layanan: **Scorekeep**
- Tipe layanan: *****
- Tuan rumah: *****
- Metode HTTP: **PUT**
- Jalur URL: **/history/***
- Sumber daya ARN: *****

Example — Tarif minimum yang lebih tinggi untuk POSTs

- Nama aturan: **POST minimum**
- Prioritas: **100**
- Waduk: **10**
- Nilai: **10** (**.1** jika dikonfigurasi menggunakan dokumen JSON)
- Nama layanan: *****
- Tipe layanan: *****
- Tuan rumah: *****

- Metode HTTP: **POST**
- Jalur URL: *
- Sumber daya ARN: *

Mengonfigurasi layanan Anda untuk menggunakan aturan pengambilan sampel

SDK X-Ray memerlukan konfigurasi tambahan untuk menggunakan aturan pengambilan sampel yang Anda konfigurasikan di konsol. Lihat topik konfigurasi untuk bahasa Anda untuk detail tentang mengonfigurasi strategi pengambilan sampel:

- Java: [Aturan pengambilan sampel](#)
- Pergi: [Aturan pengambilan sampel](#)
- Node.js: [Aturan pengambilan sampel](#)
- Python: [Aturan pengambilan sampel](#)
- Ruby: [Aturan pengambilan sampel](#)
- .NET: [Aturan pengambilan sampel](#)

Untuk API Gateway, lihat [Dukungan penelusuran aktif Amazon API Gateway untuk AWS X-Ray](#).

Melihat hasil pengambilan sampel

Halaman Pengambilan sampel konsol X-Ray menampilkan informasi mendetail tentang cara layanan Anda menggunakan setiap aturan pengambilan sampel.

Kolom Tren menunjukkan cara aturan telah digunakan dalam beberapa menit terakhir. Setiap kolom menunjukkan statistik untuk jendela 10 detik.

Statistik pengambilan sampel

- Total aturan yang cocok: Jumlah permintaan yang cocok dengan aturan ini. Jumlah ini tidak termasuk permintaan yang mungkin cocok dengan aturan ini, tetapi cocok dengan aturan dengan prioritas lebih tinggi terlebih dahulu.
- Total sampel: Jumlah permintaan yang direkam.
- Sampel dengan tarif tetap: Jumlah permintaan yang diambil sampelnya dengan menerapkan tarif tetap aturan.

- Sampel dengan batas reservoir: Jumlah permintaan sampel menggunakan kuota yang ditetapkan oleh X-Ray.
- Dipinjam dari waduk: Jumlah permintaan yang diambil sampelnya dengan meminjam dari reservoir. Saat pertama kali layanan mencocokkan permintaan dengan aturan, kuota belum ditetapkan oleh X-Ray. Namun, jika reservoir setidaknya 1, layanan meminjam satu pelacakan per detik hingga X-Ray menetapkan kuota.

Untuk informasi selengkapnya tentang statistik pengambilan sampel dan cara layanan menggunakan aturan pengambilan sampel, lihat [Penggunaan aturan pengambilan sampel dengan API X-Ray](#).

Langkah selanjutnya

Anda dapat menggunakan API X-Ray untuk mengelola aturan pengambilan sampel. Dengan API, Anda dapat membuat dan memperbarui aturan secara terprogram sesuai jadwal, atau sebagai respons terhadap alarm atau notifikasi. Lihat [Mengkonfigurasi pengaturan sampling, grup, dan enkripsi dengan API AWS X-Ray](#) untuk instruksi dan contoh aturan tambahan.

X-Ray SDK dan Layanan AWS juga menggunakan X-Ray API untuk membaca aturan sampling, melaporkan hasil sampling, dan mendapatkan target sampling. Layanan harus melacak intensitas layanan dalam menerapkan setiap aturan, mengevaluasi aturan berdasarkan prioritas, dan meminjam dari reservoir saat permintaan cocok dengan aturan ketika X-Ray belum menetapkan kuota layanan. Untuk detail selengkapnya tentang cara layanan menggunakan API untuk pengambilan sampel, lihat [Penggunaan aturan pengambilan sampel dengan API X-Ray](#).

Ketika X-Ray SDK memanggil sampling APIs, ia menggunakan daemon X-Ray sebagai proxy. Jika Anda sudah menggunakan TCP port 2000, Anda dapat mengonfigurasi daemon untuk menjalankan proksi pada port yang berbeda. Lihat [Mengkonfigurasi daemon AWS X-Ray](#) untuk rincian selengkapnya.

Konsol deep linking

Anda dapat menggunakan rute dan kueri untuk menautkan jauh ke jejak tertentu, atau tampilan jejak dan peta jejak yang difilter.

laman konsol

- Halaman selamat datang - [xray/home#/welcome](#)
- Memulai — [xray/home#/getting-mulai](#)

- Peta jejak — [xray/home#/service-map](#)
- Jejak — [xray/home#/traces](#)

Pelacakan

Anda dapat membuat tautan untuk timeline, raw, dan peta tampilan pelacakan individu.

Lacak timeline – `xray/home#/traces/trace-id`

Data pelacakan mentah – `xray/home#/traces/trace-id/raw`

Example Data pelacakan mentah –

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

Ekspresi Filter

Tautan ke daftar pelacakan yang difilter.

Tampilan pelacakan yang difilter – `xray/home#/traces?filter=filter-expression`

Example – ekspresi filter

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

Example – ekspresi filter (URL dikodekan)

```
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
```

Untuk informasi selengkapnya tentang ekspresi filter, lihat [Menggunakan ekspresi filter](#).

Rentang waktu

Tentukan lamanya waktu atau waktu mulai dan berakhir dalam format ISO86 01. Rentang waktu berada dalam UTC dan dapat mencapai 6 jam.

Lamanya waktu – `xray/home#/page?timeRange=range-in-minutes`

Example — jejak peta selama satu jam terakhir

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

Waktu mulai dan akhir – xray/home#/page?timeRange=*start~end*

Example – rentang waktu yang akurat untuk detik

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

Example – rentang waktu yang akurat untuk menit

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

Wilayah

Tentukan tautan Wilayah AWS ke halaman di Wilayah itu. Jika Anda tidak menentukan Wilayah, konsol tersebut akan mengalihkan Anda ke Wilayah terakhir yang dikunjungi.

Wilayah — xray/home?**region=region**#/page

Example — peta jejak di AS Barat (Oregon) (us-west-2)

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

Saat Anda menyertakan Region dengan parameter kueri lainnya, kueri Region berjalan sebelum hash, dan X-Ray-specific kueri akan mengikuti nama halaman.

Example — jejak peta untuk satu jam terakhir di US West (Oregon) (us-west-2)

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

Gabungan

Example – pelacakan terbaru dengan filter durasi

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

Output

- Halaman – Pelacakan
- Rentang Waktu – 15 menit terakhir
- Filter - durasi ≥ 5 DAN durasi ≤ 8

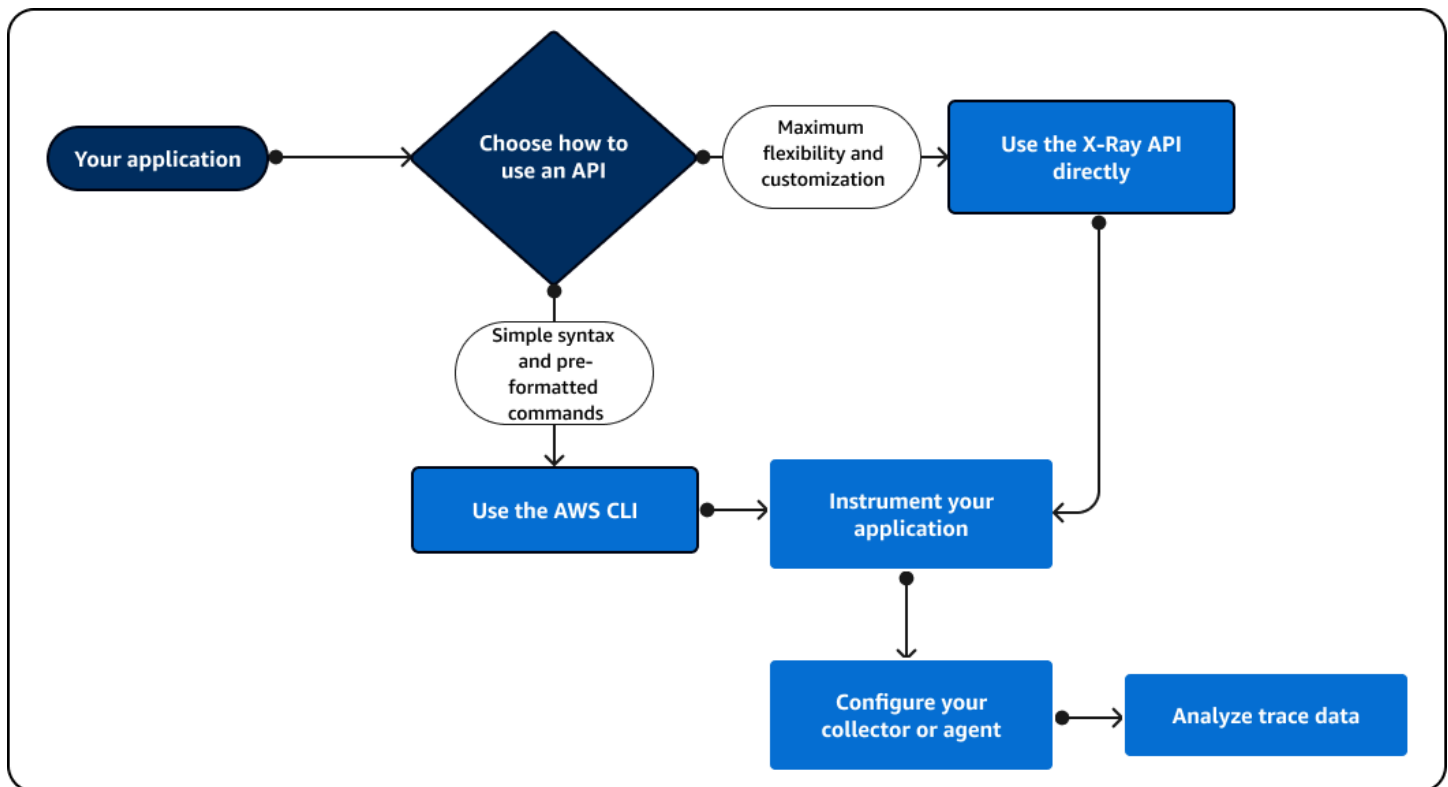
Gunakan API X-Ray

Jika X-Ray SDK tidak mendukung bahasa pemrograman Anda, Anda dapat menggunakan X-Ray APIs secara langsung atau AWS Command Line Interface (AWS CLI) untuk memanggil perintah X-Ray API. Gunakan panduan berikut untuk memilih cara Anda berinteraksi dengan API:

- Gunakan sintaks AWS CLI untuk lebih sederhana menggunakan perintah yang telah diformat sebelumnya atau dengan opsi di dalam permintaan Anda.
- Gunakan X-Ray API secara langsung untuk fleksibilitas maksimum dan kustomisasi untuk permintaan yang Anda buat ke X-Ray.

Jika Anda menggunakan [X-Ray API](#) secara langsung alih-alih AWS CLI, Anda harus membuat parameter permintaan Anda dalam format data yang benar dan mungkin juga harus mengonfigurasi otentikasi dan penanganan kesalahan.

Diagram berikut menunjukkan panduan untuk memilih cara berinteraksi dengan X-Ray API:



Gunakan X-Ray API untuk mengirim data jejak langsung ke X-Ray. X-Ray API mendukung semua fungsi yang tersedia di X-Ray SDK termasuk tindakan umum berikut:

- [PutTraceSegments](#)— Mengunggah dokumen segmen ke X-Ray.
- [BatchGetTraces](#)— Mengambil daftar jejak dalam daftar jejak IDs. Setiap jejak yang diambil adalah kumpulan dokumen segmen dari satu permintaan.
- [GetTraceSummaries](#)— Mengambil IDs dan anotasi untuk jejak. Anda dapat menentukan a `FilterExpression` untuk mengambil subset ringkasan jejak.
- [GetTraceGraph](#)— Mengambil grafik layanan untuk ID jejak tertentu.
- [GetServiceGraph](#)— Mengambil a JSON dokumen diformat yang menjelaskan layanan yang memproses permintaan masuk dan memanggil permintaan hilir.

Anda juga dapat menggunakan AWS Command Line Interface (AWS CLI) di dalam kode aplikasi Anda untuk berinteraksi secara terprogram dengan X-Ray. AWS CLI Mendukung semua fungsi yang tersedia di X-Ray SDK termasuk yang lain Layanan AWS. Fungsi-fungsi berikut adalah versi operasi API yang terdaftar sebelumnya dengan format yang lebih sederhana:

- [put-trace-segments](#)— Mengunggah dokumen segmen ke X-Ray.

- [batch-get-traces](#)— Mengambil daftar jejak dalam daftar jejak IDs. Setiap jejak yang diambil adalah kumpulan dokumen segmen dari satu permintaan.
- [get-trace-summaries](#)— Mengambil IDs dan anotasi untuk jejak. Anda dapat menentukan a `FilterExpression` untuk mengambil subset ringkasan jejak.
- [get-trace-graph](#)— Mengambil grafik layanan untuk ID jejak tertentu.
- [get-service-graph](#)— Mengambil dokumen yang JSON diformat yang menjelaskan layanan yang memproses permintaan masuk dan memanggil permintaan hilir.

Untuk memulai, Anda harus menginstal [AWS CLI](#) untuk sistem operasi Anda. AWS mendukung Linux, macOS and Windows sistem operasi. Untuk informasi selengkapnya tentang daftar perintah X-Ray, lihat [panduan Referensi AWS CLI Perintah untuk X-Ray](#).

Topik

- [Menggunakan AWS X-Ray API dengan AWS CLI](#)
- [Mengirim data pelacakan ke AWS X-Ray](#)
- [Mendapatkan data dari AWS X-Ray](#)
- [Mengkonfigurasi pengaturan sampling, grup, dan enkripsi dengan API AWS X-Ray](#)
- [Penggunaan aturan pengambilan sampel dengan API X-Ray](#)
- [AWS X-Ray dokumen segmen](#)

Menggunakan AWS X-Ray API dengan AWS CLI

AWS CLI memungkinkan Anda mengakses layanan X-Ray secara langsung dan menggunakan yang sama dengan APIs yang digunakan konsol X-Ray untuk mengambil grafik layanan dan data jejak mentah. Aplikasi sampel mencakup skrip yang menunjukkan cara menggunakannya APIs dengan AWS CLI.

Prasyarat

Tutorial ini menggunakan aplikasi sampel Scorekeep dan termasuk skrip untuk menghasilkan data pelacakan dan peta layanan. Ikuti instruksi di [Memulai tutorial](#) untuk meluncurkan aplikasi.

Tutorial ini menggunakan AWS CLI untuk menunjukkan penggunaan dasar X-Ray API. AWS CLI, [tersedia untuk Windows, Linux, dan OS-X](#), menyediakan akses baris perintah ke publik untuk semua APIs Layanan AWS

Note

Anda harus memverifikasi bahwa Anda AWS CLI dikonfigurasi ke Wilayah yang sama tempat aplikasi sampel Scorekeep Anda dibuat.

Skrip disertakan untuk menguji aplikasi sampel menggunakan cURL untuk mengirim lalu lintas ke API dan jq untuk mengurai output. Anda dapat mengunduh jq eksekusi dari stedolan.github.io, dan curl eksekusi dari <https://curl.haxx.se/download.html>. Sebagian besar instalasi Linux dan OS X mencakup cURL.

Hasilkan data pelacakan

Aplikasi web terus menghasilkan lalu lintas ke API setiap beberapa detik saat game sedang berlangsung, tetapi hanya menghasilkan satu tipe permintaan. Gunakan skrip `test-api.sh` untuk menjalankan skenario end to end dan menghasilkan data pelacakan yang lebih beragam saat Anda menguji API.

Untuk menggunakan skrip **test-api.sh**

1. Buka [Konsol Elastic Beanstalk](#).
2. Navigasikan ke [konsol manajemen](#) untuk lingkungan Anda.
3. Salin URL lingkungan dari header halaman.
4. Buka `bin/test-api.sh` dan ganti nilai untuk API dengan URL lingkungan Anda.

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

5. Jalankan skrip untuk menghasilkan lalu lintas ke API.

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

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

Gunakan API X-Ray

AWS CLI menyediakan perintah untuk semua tindakan API yang disediakan X-Ray, termasuk [GetServiceGraph](#) dan [GetTraceSummaries](#). Lihat [AWS X-Ray Referensi API](#) untuk informasi selengkapnya tentang semua tindakan yang didukung dan tipe data yang mereka gunakan.

Example bin/service-graph.sh

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

Skip mengambil grafik layanan selama 10 menit terakhir.

```
~/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'

```

Script mengambil jejak yang URLs dihasilkan antara satu dan dua menit yang lalu.

```

~/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[*]'
```

Skip mengambil pelacakan penuh yang dihasilkan antara satu dan dua menit yang lalu.

```
~/eb-java-scorekeep$ ./bin/full-traces.sh | less
[
  {
    "Segments": [
      {
        "Id": "3f212bc237bafd5d",
        "Document": "{\"id\":\"3f212bc237bafd5d\",\"name\":\"DynamoDB\",
        \"trace_id\":\"1-5828d9f2-a90669393f4343211bc1cf75\",\"start_time\":1.479072242459E9,
        \"end_time\":1.479072242477E9,\"parent_id\":\"72a08dcf87991ca9\",\"http\":
        {\"response\":{\"content_length\":60,\"status\":200}},\"inferred\":true,\"aws\":
        {\"consistent_read\":false,\"table_name\":\"scorekeep-session-xray\",\"operation\":
        \"GetItem\",\"request_id\":\"QAKE0S8DD0LJM245KAOPMA746BVV4KQNS05AEMVJF66Q9ASUAAJG\",
        \"resource_names\":[\"scorekeep-session-xray\"]},\"origin\":\"AWS::DynamoDB::Table\"}"
      },
      {
        "Id": "309e355f1148347f",
        "Document": "{\"id\":\"309e355f1148347f\",\"name\":\"DynamoDB\",
        \"trace_id\":\"1-5828d9f2-a90669393f4343211bc1cf75\",\"start_time\":1.479072242477E9,
        \"end_time\":1.479072242494E9,\"parent_id\":\"37f14ef837f00022\",\"http\":
        {\"response\":{\"content_length\":606,\"status\":200}},\"inferred\":true,\"aws\":
        {\"table_name\":\"scorekeep-game-xray\",\"operation\":\"UpdateItem\",\"request_id
        \":\"388GER0C4PCA6D59ED3CTI5EEJVV4KQNS05AEMVJF66Q9ASUAAJG\", \"resource_names\":
        [\"scorekeep-game-xray\"]},\"origin\":\"AWS::DynamoDB::Table\"}"
      }
    ],
    "Id": "1-5828d9f2-a90669393f4343211bc1cf75",
    "Duration": 0.05099987983703613
  }
  ...
```

Pembersihan

Hentikan lingkungan Elastic Beanstalk Anda untuk mematikan instans EC2 Amazon, tabel DynamoDB, dan sumber daya lainnya.

Untuk mengakhiri lingkungan Elastic Beanstalk

1. Buka [Konsol Elastic Beanstalk](#).

2. Navigasikan ke [konsol manajemen](#) untuk lingkungan Anda.
3. Pilih Tindakan.
4. Pilih Akhiri Lingkungan.
5. Pilih Akhiri.

Data pelacakan dihapus secara otomatis dari X-Ray setelah 30 hari.

Mengirim data pelacakan ke AWS X-Ray

Anda dapat mengirim data pelacakan ke X-Ray dalam bentuk dokumen segmen. Sebuah dokumen segmen adalah string berformat JSON yang berisi informasi tentang pekerjaan yang dilakukan aplikasi Anda dalam pelayanan permintaan. Aplikasi Anda dapat mencatat data tentang pekerjaan yang dilakukannya sendiri di segmen, atau pekerjaan yang menggunakan layanan hilir dan sumber daya di subsegment.

Segmen mencatat informasi tentang pekerjaan yang dilakukan aplikasi Anda. Segmen, minimal, mencatat waktu yang dihabiskan untuk tugas, nama, dan dua IDs. ID pelacakan melacak permintaan saat perjalanan antara layanan. ID segmen melacak pekerjaan yang dilakukan untuk permintaan oleh satu layanan.

Example Segmen lengkap minimal

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

Ketika permintaan diterima, Anda dapat mengirim segmen yang sedang berlangsung sebagai placeholder sampai permintaan selesai.

Example Segmen yang sedang berlangsung

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

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

Anda dapat mengirim segmen ke X-Ray secara langsung, dengan [PutTraceSegments](#), atau [melalui daemon X-Ray](#).

Sebagian besar aplikasi memanggil layanan lain atau mengakses sumber daya dengan AWS SDK. Mencatat informasi tentang panggilan hilir di subsegmen. X-Ray menggunakan subsegmen untuk mengidentifikasi layanan hilir yang tidak mengirim segmen dan membuat entri untuk mereka di grafik layanan.

Sebuah subsegmen dapat ditanamkan dalam dokumen segmen penuh, atau dikirim secara terpisah. Kirim subsegmen secara terpisah untuk melacak panggilan hilir secara asinkron untuk permintaan yang berlangsung lama, atau untuk menghindari melebihi ukuran dokumen segmen maksimum (64 kB).

Example Subsegmen

Subsegmen memiliki type dari subsegment dan parent_id yang mengidentifikasi segmen induk.

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

Untuk informasi selengkapnya tentang bidang dan nilai yang dapat Anda sertakan dalam segmen dan subsegmen, lihat [AWS X-Ray dokumen segmen](#).

Bagian-bagian

- [Menghasilkan jejak IDs](#)
- [Menggunakan PutTraceSegments](#)
- [Mengirim dokumen segmen ke daemon X-Ray](#)

Menghasilkan jejak IDs

Untuk mengirim data ke X-Ray, Anda harus menghasilkan ID jejak unik untuk setiap permintaan.

Format ID jejak X-Ray

`trace_id`X-Ray terdiri dari tiga angka yang dipisahkan oleh tanda hubung. Contohnya, `1-58406520-a006649127e371903a2de979`. Hal ini mencakup:

- Nomor versi, yaitu `1`.
- Waktu permintaan asli dalam waktu epoch Unix menggunakan 8 digit heksadesimal.

Misalnya, 10:00 AM 1 Desember 2016 PST dalam waktu epoch adalah `1480615200` detik atau `58406520` dalam digit heksadesimal.

- Pengidentifikasi 96-bit yang unik secara global untuk jejak dalam 24 digit heksadesimal.

Note

X-Ray sekarang mendukung jejak IDs yang dibuat menggunakan OpenTelemetry dan kerangka kerja lain yang sesuai dengan spesifikasi [W3C Trace](#) Context. ID jejak W3C harus diformat dalam format X-Ray Trace ID saat mengirim ke X-Ray. Misalnya, ID jejak W3C `4efaaf4d1e8720b39541901950019ee5` harus diformat seperti `1-4efaaf4d-1e8720b39541901950019ee5` saat mengirim ke X-Ray. Jejak X-Ray IDs menyertakan cap waktu permintaan asli dalam waktu epoch Unix, tetapi ini tidak diperlukan saat mengirim jejak IDs W3C dalam format X-Ray.

Anda dapat menulis skrip untuk menghasilkan jejak X-Ray IDs untuk pengujian. Berikut ini adalah dua contoh.

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"
```

Lihat contoh aplikasi Scorekeep untuk skrip yang membuat jejak IDs dan mengirim segmen ke daemon X-Ray.

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

Menggunakan PutTraceSegments

Anda dapat mengunggah dokumen segmen dengan API [PutTraceSegments](#). API memiliki parameter tunggal, `TraceSegmentDocuments`, yang mengambil daftar dokumen segmen JSON.

Dengan AWS CLI, gunakan perintah `aws xray put-trace-segments` untuk mengirim dokumen segmen langsung ke 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 dan Windows PowerShell memiliki persyaratan yang berbeda untuk mengutip dan melarikan diri dari kutipan dalam string JSON. Lihat [Mengutip String](#) di Panduan Pengguna AWS CLI untuk detail.

Output mencantumkan segmen yang gagal diproses. Misalnya, jika tanggal di ID pelacakan terlalu lama di masa lalu, Anda melihat kesalahan seperti berikut ini.

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

Anda dapat melewati beberapa dokumen segmen pada saat yang sama, dipisahkan oleh spasi.

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

Mengirim dokumen segmen ke daemon X-Ray

Daripada mengirim dokumen segmen ke API X-Ray, Anda dapat mengirim segmen dan subsegmen ke daemon X-Ray, yang akan menyangga dokumen tersebut dan mengunggahnya ke API X-Ray dalam batch. X-Ray SDK mengirimkan dokumen segmen ke daemon untuk menghindari panggilan ke AWS secara langsung.

Note

Lihat [Menjalankan daemon X-Ray secara lokal](#) sebagai petunjuk tentang menjalankan daemon.

Kirim segmen di JSON melalui UDP port 2000, diawali dengan header 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"}
```

Di Linux, Anda dapat mengirim dokumen segmen ke daemon dari terminal Bash. Simpan header dan segmen dokumen ke file teks dan alirkan ke /dev/udp dengan 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"}
```

Periksa [log daemon](#) untuk memverifikasi bahwa segmen tersebut telah dikirim ke 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)
```

Mendapatkan data dari AWS X-Ray

AWS X-Ray memproses data jejak yang Anda kirim untuk menghasilkan jejak lengkap, ringkasan jejak, dan grafik layanan di JSON. Anda dapat mengambil data yang dihasilkan langsung dari API dengan AWS CLI.

Bagian-bagian

- [Mengambil grafik layanan](#)
- [Mengambil grafik layanan berdasarkan grup](#)
- [Mengambil pelacakan](#)
- [Mengambil dan menyempurnakan analitik akar masalah](#)

Mengambil grafik layanan

Anda dapat menggunakan [GetServiceGraph](#) API untuk mengambil grafik layanan JSON. API memerlukan waktu mulai dan waktu berakhir, yang dapat Anda hitung dari terminal Linux dengan perintah `date`.

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

`date +%s` mencetak tanggal dalam hitungan detik. Gunakan angka ini sebagai waktu berakhir dan kurangi waktunya untuk mendapatkan waktu mulai.

Example Skrip untuk mengambil grafik layanan selama 10 menit terakhir

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

Contoh berikut menunjukkan grafik layanan dengan 4 node, termasuk node klien, EC2 instance, tabel DynamoDB, dan topik Amazon SNS.

Example GetServiceGraph keluaran

```
{
  "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
        }
    ]
}

```

Mengambil grafik layanan berdasarkan grup

Untuk memanggil grafik layanan berdasarkan konten grup, sertakan groupName atau groupARN. Contoh berikut menunjukkan panggilan grafik layanan ke grup bernama Example1.

Example Skrip untuk mengambil grafik layanan berdasarkan nama untuk grup Example1

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

Mengambil pelacakan

Anda dapat menggunakan API [GetTraceSummaries](#) untuk mendapatkan daftar ringkasan pelacakan. Ringkasan jejak mencakup informasi yang dapat Anda gunakan untuk mengidentifikasi jejak yang ingin Anda unduh secara lengkap, termasuk anotasi, informasi permintaan dan respons, dan IDs

Ada dua bendera TimeRangeType tersedia ketika memanggil `aws xray get-trace-summaries`:

- **TraceId**— `GetTraceSummaries` Pencarian default menggunakan waktu traceID dan mengembalikan jejak yang dimulai dalam rentang yang dihitung. `[start_time, end_time)` Rentang stempel waktu ini dihitung berdasarkan pengkodean stempel waktu di dalam TraceId, atau dapat didefinisikan secara manual.

- Waktu peristiwa – Untuk mencari peristiwa yang terjadi dari waktu ke waktu, AWS X-Ray mengizinkan pencarian pelacakan menggunakan stempel waktu peristiwa. Waktu peristiwa mengembalikan pelacakan yang aktif selama jangkauan [start_time, end_time), terlepas dari kapan penelusuran dimulai.

Gunakan perintah `aws xray get-trace-summaries` untuk mendapatkan daftar ringkasan pelacakan. Perintah berikut mendapatkan daftar ringkasan jejak dari antara 1 dan 2 menit di masa lalu menggunakan waktu default `TraceId`.

Example Skrip untuk mendapatkan ringkasan pelacakan

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

Example `GetTraceSummaries` keluaran

```
{
  "TraceSummaries": [
    {
      "HasError": false,
      "Http": {
        "HttpStatus": 200,
        "ClientId": "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,
        "ClientId": "205.255.255.183",
```

```

        "HttpURL": "http://scorekeep.elasticbeanstalk.com/api/user",
        "UserAgent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36",
        "HttpMethod": "POST"
    },
    "Users": [
        {
            "UserName": "5M388M1E"
        }
    ],
    "HasFault": false,
    "Annotations": {
        "UserID": [
            {
                "AnnotationValue": {
                    "StringValue": "5M388M1E"
                }
            }
        ],
        "Name": [
            {
                "AnnotationValue": {
                    "StringValue": "01a"
                }
            }
        ]
    },
    "ResponseTime": 3.232,
    "Duration": 3.232,
    "Id": "1-59602603-23fc5b688855d396af79b496",
    "HasThrottle": false
    }
],
"ApproximateTime": 1499473304.0,
"TracesProcessedCount": 2
}

```

Gunakan ID pelacakan dari output untuk mengambil pelacakan penuh dengan API [BatchGetTraces](#).

Example BatchGetTraces perintah

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

Example BatchGetTraces keluaran

```
{
  "Traces": [
    {
      "Duration": 3.232,
      "Segments": [
        {
          "Document": "{\"id\":\"1fb07842d944e714\",\"name\": \"random-name\", \"start_time\":1.499473411677E9, \"end_time\":1.499473414572E9, \"parent_id\":\"0c544c1b1bbff948\", \"http\":{\"response\":{\"status\":200}}, \"aws\":{\"request_id\":\"ac086670-6373-11e7-a174-f31b3397f190\"}, \"trace_id\":\"1-59602603-23fc5b688855d396af79b496\", \"origin\":\"AWS::Lambda\", \"resource_arn\":\"arn:aws:lambda:us-west-2:123456789012:function:random-name\"}\",
          "Id": "1fb07842d944e714"
        },
        {
          "Document": "{\"id\":\"194fcc8747581230\",\"name\":\"Scorekeep\", \"start_time\":1.499473411562E9, \"end_time\":1.499473414794E9, \"http\":{\"request\":{\"url\":\"http://scorekeep.elasticbeanstalk.com/api/user\", \"method\":\"POST\", \"user_agent\":\"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36\", \"client_ip\":\"205.251.233.183\"}, \"response\":{\"status\":200}}, \"aws\":{\"elastic_beanstalk\":{\"version_label\":\"app-abb9-170708_002045\", \"deployment_id\":406, \"environment_name\":\"scorekeep-dev\", \"ec2\":{\"availability_zone\":\"us-west-2c\", \"instance_id\":\"i-0cd9e448944061b4a\"}, \"xray\":{\"sdk_version\":\"1.1.2\", \"sdk\":\"X-Ray for Java\"}}, \"service\":{\"}, \"trace_id\":\"1-59602603-23fc5b688855d396af79b496\", \"user\":\"5M388M1E\", \"origin\":\"AWS::ElasticBeanstalk::Environment\", \"subsegments\":[{\"id\":\"0c544c1b1bbff948\", \"name\":\"Lambda\", \"start_time\":1.499473411629E9, \"end_time\":1.499473414572E9, \"http\":{\"response\":{\"status\":200, \"content_length\":14}}, \"aws\":{\"log_type\":\"None\", \"status_code\":200, \"function_name\":\"random-name\", \"invocation_type\":\"RequestResponse\", \"operation\":\"Invoke\", \"request_id\":\"ac086670-6373-11e7-a174-f31b3397f190\", \"resource_names\":[\"random-name\"]}, \"namespace\":\"aws\"}, {\"id\":\"071684f2e555e571\", \"name\":\"## UserModel.saveUser\", \"start_time\":1.499473414581E9, \"end_time\":1.499473414769E9, \"metadata\":{\"debug\":{\"test\":\"Metadata string from UserModel.saveUser\"}}, \"subsegments\":[{\"id\":\"4cd3f10b76c624b4\", \"name\":\"DynamoDB\", \"start_time\":1.49947341469E9, \"end_time\":1.499473414769E9, \"http\":{\"response\":{\"status\":200, \"content_length\":57}}, \"aws\":{\"table_name\":\"scorekeep-user\", \"operation\":\"UpdateItem\", \"request_id\":\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738B0B4KQNS05AEMVJF66Q9\", \"resource_names\":[\"scorekeep-user\"]}, \"namespace\":\"aws\"}]}}\",
          "Id": "194fcc8747581230"
        },
        {
          "Document": "{\"id\":\"194fcc8747581230\", \"start_time\":1.499473411562E9, \"end_time\":1.499473414794E9, \"http\":{\"request\":{\"url\":\"http://scorekeep.elasticbeanstalk.com/api/user\", \"method\":\"POST\", \"user_agent\":\"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36\", \"client_ip\":\"205.251.233.183\"}, \"response\":{\"status\":200}}, \"aws\":{\"elastic_beanstalk\":{\"version_label\":\"app-abb9-170708_002045\", \"deployment_id\":406, \"environment_name\":\"scorekeep-dev\", \"ec2\":{\"availability_zone\":\"us-west-2c\", \"instance_id\":\"i-0cd9e448944061b4a\"}, \"xray\":{\"sdk_version\":\"1.1.2\", \"sdk\":\"X-Ray for Java\"}}, \"service\":{\"}, \"trace_id\":\"1-59602603-23fc5b688855d396af79b496\", \"user\":\"5M388M1E\", \"origin\":\"AWS::ElasticBeanstalk::Environment\", \"subsegments\":[{\"id\":\"0c544c1b1bbff948\", \"name\":\"Lambda\", \"start_time\":1.499473411629E9, \"end_time\":1.499473414572E9, \"http\":{\"response\":{\"status\":200, \"content_length\":14}}, \"aws\":{\"log_type\":\"None\", \"status_code\":200, \"function_name\":\"random-name\", \"invocation_type\":\"RequestResponse\", \"operation\":\"Invoke\", \"request_id\":\"ac086670-6373-11e7-a174-f31b3397f190\", \"resource_names\":[\"random-name\"]}, \"namespace\":\"aws\"}, {\"id\":\"071684f2e555e571\", \"name\":\"## UserModel.saveUser\", \"start_time\":1.499473414581E9, \"end_time\":1.499473414769E9, \"metadata\":{\"debug\":{\"test\":\"Metadata string from UserModel.saveUser\"}}, \"subsegments\":[{\"id\":\"4cd3f10b76c624b4\", \"name\":\"DynamoDB\", \"start_time\":1.49947341469E9, \"end_time\":1.499473414769E9, \"http\":{\"response\":{\"status\":200, \"content_length\":57}}, \"aws\":{\"table_name\":\"scorekeep-user\", \"operation\":\"UpdateItem\", \"request_id\":\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738B0B4KQNS05AEMVJF66Q9\", \"resource_names\":[\"scorekeep-user\"]}, \"namespace\":\"aws\"}]}}\",
          "Id": "194fcc8747581230"
        }
      ]
    }
  ]
}
```

```

        "Document": "{\\"id\\":\\"00f91aa01f4984fd\\",\\"name\\":
        \\"random-name\\",\\"start_time\\":1.49947341283E9,\\"end_time\\":1.49947341457E9,
        \\"parent_id\\":\\"1fb07842d944e714\\",\\"aws\\":{\\"function_arn\\":\\"arn:aws:lambda:us-
        west-2:123456789012:function:random-name\\",\\"resource_names\\":[\\"random-name\\"],
        \\"account_id\\":\\"123456789012\\"},\\"trace_id\\":\\"1-59602603-23fc5b688855d396af79b496\\",
        \\"origin\\":\\"AWS::Lambda::Function\\",\\"subsegments\\":[\\"{\\"id\\":\\"e6d2fe619f827804\\",
        \\"name\\":\\"annotations\\",\\"start_time\\":1.499473413012E9,\\"end_time\\":1.499473413069E9,
        \\"annotations\\":{\\"UserID\\":\\"5M388M1E\\",\\"Name\\":\\"0la\\"}},{\\"id\\":\\"b29b548af4d54a0f
        \\",\\"name\\":\\"SNS\\",\\"start_time\\":1.499473413112E9,\\"end_time\\":1.499473414071E9,
        \\"http\\":{\\"response\\":{\\"status\\":200}}},\\"aws\\":{\\"operation\\":\\"Publish\\",
        \\"region\\":\\"us-west-2\\",\\"request_id\\":\\"a2137970-f6fc-5029-83e8-28aadeb99198\\",
        \\"retries\\":0,\\"topic_arn\\":\\"arn:aws:sns:us-west-2:123456789012:awseb-e-
        ruag3jyweb-stack-NotificationTopic-6B829NT9V509\\"},\\"namespace\\":\\"aws\\"},{\\"id\\":
        \\"2279c0030c955e52\\",\\"name\\":\\"Initialization\\",\\"start_time\\":1.499473412064E9,
        \\"end_time\\":1.499473412819E9,\\"aws\\":{\\"function_arn\\":\\"arn:aws:lambda:us-
        west-2:123456789012:function:random-name\\"}}]}",
        "Id": "00f91aa01f4984fd"
    },
    {
        "Document": "{\\"id\\":\\"17ba309b32c7fbaf\\",\\"name\\":
        \\"DynamoDB\\",\\"start_time\\":1.49947341469E9,\\"end_time\\":1.499473414769E9,
        \\"parent_id\\":\\"4cd3f10b76c624b4\\",\\"inferred\\":true,\\"http\\":{\\"response
        \\":{\\"status\\":200,\\"content_length\\":57}},\\"aws\\":{\\"table_name
        \\":\\"scorekeep-user\\",\\"operation\\":\\"UpdateItem\\",\\"request_id\\":
        \\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738B0B4KQNS05AEMVJF66Q9\\",\\"resource_names\\":
        [\\"scorekeep-user\\"],\\"trace_id\\":\\"1-59602603-23fc5b688855d396af79b496\\",\\"origin\\":
        \\"AWS::DynamoDB::Table\\"}",
        "Id": "17ba309b32c7fbaf"
    },
    {
        "Document": "{\\"id\\":\\"1ee3c4a523f89ca5\\",\\"name\\":\\"SNS
        \\",\\"start_time\\":1.499473413112E9,\\"end_time\\":1.499473414071E9,\\"parent_id\\":
        \\"b29b548af4d54a0f\\",\\"inferred\\":true,\\"http\\":{\\"response\\":{\\"status\\":200}},\\"aws
        \\":{\\"operation\\":\\"Publish\\",\\"region\\":\\"us-west-2\\",\\"request_id\\":\\"a2137970-
        f6fc-5029-83e8-28aadeb99198\\",\\"retries\\":0,\\"topic_arn\\":\\"arn:aws:sns:us-
        west-2:123456789012:awseb-e-ruag3jyweb-stack-NotificationTopic-6B829NT9V509\\"},
        \\"trace_id\\":\\"1-59602603-23fc5b688855d396af79b496\\",\\"origin\\":\\"AWS::SNS\\"}",
        "Id": "1ee3c4a523f89ca5"
    }
  ],
  "Id": "1-59602603-23fc5b688855d396af79b496"
}
],
"UnprocessedTraceIds": []

```

```
}
```

Pelacakan lengkap mencakup dokumen untuk setiap segmen, dikompilasi dari semua dokumen segmen yang diterima dengan ID pelacakan yang sama. Dokumen-dokumen ini tidak mewakili data karena dikirim ke X-Ray oleh aplikasi Anda. Sebaliknya, data-data tersebut mewakili dokumen yang diproses dihasilkan oleh layanan X-Ray. X-Ray membuat dokumen pelacakan lengkap dengan mengompilasi dokumen segmen yang dikirim oleh aplikasi Anda, dan menghapus data yang tidak sesuai dengan [skema dokumen segmen](#).

X-Ray juga membuat segmen yang disimpulkan untuk panggilan hilir ke layanan yang tidak mengirim segmen itu sendiri. Misalnya, saat Anda memanggil DynamoDB dengan klien berinstrumen, X-Ray SDK mencatat subsegmen dengan detail tentang panggilan dari sudut pandangnya. Namun, DynamoDB tidak mengirim segmen yang sesuai. X-Ray menggunakan informasi di subsegmen untuk membuat segmen yang disimpulkan untuk mewakili sumber daya DynamoDB di peta jejak, dan menambahkannya ke dokumen jejak.

Untuk mendapatkan beberapa jejak dari API, Anda memerlukan daftar jejak IDs, yang dapat Anda ekstrak dari output `get-trace-summaries` dengan [AWS CLI kueri](#). Alihkan ulang daftar ke input `batch-get-traces` untuk mendapatkan pelacakan penuh untuk jangka waktu tertentu.

Example Skrip untuk mendapatkan pelacakan penuh selama satu menit

```
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[*]'
```

Mengambil dan menyempurnakan analitik akar masalah

Setelah membuat ringkasan jejak dengan [GetTraceSummaries API](#), ringkasan jejak sebagian dapat digunakan kembali dalam format JSON mereka untuk membuat ekspresi filter yang disempurnakan berdasarkan akar penyebab. Lihat contoh di bawah untuk panduan langkah-langkah penyempurnaan.

Example Contoh GetTraceSummaries keluaran - bagian akar penyebab waktu respons

```
{
  "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
      }
    ]
  }
]
}

```

Dengan mengedit dan menghilangkan output di atas, JSON ini dapat menjadi filter untuk entitas akar masalah yang cocok. Untuk setiap bidang yang ada di JSON, setiap kandidat yang cocok harus sama persis, atau pelacakan tidak akan dikembalikan. Bidang yang dihapus menjadi nilai wildcard, format yang kompatibel dengan struktur kueri ekspresi filter.

Example Akar masalah waktu respons yang diformat ulang

```

{
  "Services": [

```

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

JSON ini kemudian digunakan sebagai bagian dari ekspresi filter melalui panggilan ke `rootcause.json = #[{}]`. Mengacu kepada [Filter Ekspresi](#) bab untuk detail selengkapnya tentang kueri dengan ekspresi filter.

Example Contoh filter JSON

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

Mengkonfigurasi pengaturan sampling, grup, dan enkripsi dengan API AWS X-Ray

AWS X-Ray menyediakan APIs untuk mengonfigurasi [aturan pengambilan sampel](#), [aturan](#) grup, dan pengaturan [enkripsi](#).

Bagian-bagian

- [Pengaturan enkripsi](#)
- [Aturan pengambilan sampel](#)

- [Grup](#)

Pengaturan enkripsi

Gunakan [PutEncryptionConfig](#) untuk menentukan kunci AWS Key Management Service (AWS KMS) yang akan digunakan untuk enkripsi.

Note

X-Ray tidak mendukung tombol KMS asimetris.

```
$ 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"
  }
}
```

Untuk ID kunci, Anda dapat menggunakan alias (seperti yang ditunjukkan dalam contoh), ID kunci, atau Amazon Resource Name (ARN).

Gunakan [GetEncryptionConfig](#) untuk mendapatkan konfigurasi saat ini. Setelah X-Ray selesai menerapkan pengaturan Anda, status akan berubah dari UPDATING ke 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"
  }
}
```

Untuk berhenti menggunakan kunci KMS dan menggunakan enkripsi default, atur jenis enkripsi ke NONE.

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

Aturan pengambilan sampel

Anda dapat mengelola [aturan pengambilan sampel](#) di akun Anda dengan X-Ray API. Untuk informasi selengkapnya tentang menambahkan dan mengelola tag, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

Dapatkan semua aturan pengambilan sampel dengan [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
    }
  ]
}
```

Aturan default berlaku untuk semua permintaan yang tidak cocok dengan aturan lain. Ini adalah aturan prioritas terendah dan tidak dapat dihapus. Namun, Anda dapat mengubah tingkat dan ukuran reservoir dengan [UpdateSamplingRule](#).

Example Input API untuk [UpdateSamplingRule](#) – 10000-default.json

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

Contoh berikut menggunakan file sebelumnya sebagai input untuk mengubah aturan default ke satu persen tanpa reservoir. Tanda adalah opsional. Jika Anda memilih untuk menambahkan tanda, kunci tanda diperlukan, dan nilai tanda adalah opsional. Untuk menghapus tag yang ada dari aturan pengambilan sampel, gunakan [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
    },
  ],
}
```

Buat aturan pengambilan sampel tambahan dengan [CreateSamplingRule](#). Bila Anda membuat aturan, sebagian besar bidang aturan diperlukan. Contoh berikut ini menampilkan pembuatan dua aturan. Aturan pertama ini menetapkan tingkat dasar untuk aplikasi sampel Scorekeep. Aturan tersebut cocok dengan semua permintaan yang dilayani oleh API yang tidak cocok dengan aturan prioritas yang lebih tinggi.

Example Input API untuk [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
  }
}
```

Aturan kedua juga berlaku untuk Scorekeep, tetapi memiliki prioritas yang lebih tinggi dan lebih spesifik. Aturan ini menetapkan tingkat pengambilan sampel yang sangat rendah atas permintaan polling. Ini adalah permintaan GET yang dibuat oleh klien setiap beberapa detik untuk memeriksa perubahan pada status game.

Example Input API untuk [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
  }
}

```

Tanda adalah opsional. Jika Anda memilih untuk menambahkan tanda, kunci tanda diperlukan, dan nilai tanda adalah opsional.

```

$ 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
}
}

```

Untuk menghapus aturan pengambilan sampel, gunakan [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
  }
}

```

Grup

Anda dapat menggunakan API X-Ray untuk mengelola grup di akun Anda. Grup adalah kumpulan pelacakan yang ditentukan oleh ekspresi filter. Anda dapat menggunakan grup untuk menghasilkan

grafik layanan tambahan dan menyediakan CloudWatch metrik Amazon. Lihat [Mendapatkan data dari AWS X-Ray](#) untuk detail selengkapnya tentang bekerja dengan grafik dan metrik layanan melalui X-Ray API. Untuk informasi selengkapnya tentang grup, lihat [Mengkonfigurasi grup](#). Untuk informasi selengkapnya tentang menambahkan dan mengelola tag, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

Buat grup dengan `CreateGroup`. Tanda adalah opsional. Jika Anda memilih untuk menambahkan tanda, kunci tanda diperlukan, dan nilai tanda adalah opsional.

```
$ 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}"
}
```

Dapatkan semua grup yang ada dengan `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"
}
```

Perbarui grup dengan `UpdateGroup`. Tanda adalah opsional. Jika Anda memilih untuk menambahkan tanda, kunci tanda diperlukan, dan nilai tanda adalah opsional. Untuk menghapus tag yang ada dari grup, gunakan [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}"
}
```

Hapus grup dengan DeleteGroup.

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

Penggunaan aturan pengambilan sampel dengan API X-Ray

AWS X-Ray SDK menggunakan X-Ray API untuk mendapatkan aturan sampling, melaporkan hasil sampling, dan mendapatkan kuota. Anda dapat menggunakan ini APIs untuk mendapatkan pemahaman yang lebih baik tentang cara kerja aturan pengambilan sampel, atau untuk menerapkan pengambilan sampel dalam bahasa yang tidak didukung oleh X-Ray SDK.

Mulailah dengan mendapatkan semua aturan pengambilan sampel menggunakan [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
  }
]
}

```

Output termasuk aturan default dan aturan kustom. Lihat [Aturan pengambilan sampel](#) jika Anda belum membuat aturan pengambilan sampel.

Mengevaluasi aturan terhadap permintaan masuk dalam urutan menaik prioritas. Ketika aturan cocok, gunakan nilai tetap dan ukuran reservoir untuk membuat keputusan pengambilan sampel. Catat permintaan sampel dan abaikan (untuk tujuan pelacakan) permintaan tanpa sampel. Hentikan evaluasi aturan ketika keputusan pengambilan sampel dibuat.

Sebuah ukuran aturan reservoir adalah jumlah target pelacakan untuk mencatat per detik sebelum menerapkan nilai tetap. Reservoir berlaku di semua layanan secara kumulatif, sehingga Anda tidak dapat menggunakannya secara langsung. Namun, jika bukan nol, Anda dapat meminjam satu pelacakan per detik dari reservoir sampai X-Ray menetapkan kuota. Sebelum menerima kuota, catat permintaan pertama setiap detik, dan terapkan nilai tetap untuk permintaan tambahan. Nilai tetapnya merupakan angka desimal antara 0 dan 1,00 (100%).

Contoh berikut menunjukkan panggilan ke [GetSamplingTargets](#) dengan detail tentang keputusan pengambilan sampel yang dibuat selama 10 detik terakhir.

```

$ 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
  }
]

```

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

Respons dari X-Ray mencakup kuota untuk digunakan daripada meminjam dari reservoir. Dalam contoh ini, layanan meminjam 10 peristiwa dari reservoir selama lebih dari 10 detik, dan menerapkan nilai tetap 10 persen untuk 100 permintaan lainnya, mengakibatkan total 20 permintaan sampel. Kuota tersebut bagus selama lima menit (ditunjukkan dengan waktu untuk tayang) atau sampai kuota baru ditetapkan. X-Ray juga dapat menetapkan interval pelaporan yang lebih lama daripada default, meskipun tidak ada di sini.

Note

Respons dari X-Ray mungkin tidak termasuk kuota saat pertama kali Anda memanggilnya. Lanjutkan peminjaman dari reservoir sampai Anda diberi kuota.

Dua bidang lainnya dalam respons mungkin menunjukkan masalah dengan input. Periksa `LastRuleModification` untuk terakhir kali Anda menelepon [GetSamplingRules](#). Jika lebih baru, dapatkan salinan aturan baru. `UnprocessedStatistics` dapat mencakup kesalahan yang menunjukkan bahwa aturan telah dihapus, dokumen statistik dalam input terlalu tua, atau izin mengalami kesalahan.

AWS X-Ray dokumen segmen

Segmen penelusuran adalah representasi JSON dari permintaan yang digunakan aplikasi Anda. Segmen pelacakan mencatat informasi tentang permintaan asli, informasi tentang pekerjaan yang dilakukan aplikasi Anda secara lokal, dan subsegmen dengan informasi tentang panggilan hilir yang dibuat aplikasi Anda ke AWS sumber daya, HTTP APIs, dan database SQL.

Sebuah dokumen segmen menyampaikan informasi tentang segmen ke X-Ray. Dokumen segmen dapat sebesar 64 kB dan berisi seluruh segmen dengan subsegment, fragmen segmen yang menunjukkan bahwa permintaan sedang berlangsung, atau subsegmen tunggal yang dikirim secara terpisah. Anda dapat mengirim dokumen segmen langsung ke X-Ray dengan menggunakan API [PutTraceSegments](#).

X-Ray mengkompilasi dan memproses dokumen segmen untuk menghasilkan ringkasan jejak yang dapat dikueri dan jejak lengkap yang dapat Anda akses dengan menggunakan dan, masing-masing. [GetTraceSummariesBatchGetTraces](#) APIs Selain segmen dan subsegmen yang Anda kirim ke X-Ray, layanan ini menggunakan informasi dalam subsegmen untuk menghasilkan segmen yang disimpulkan serta menambahkan segmen tersebut ke penelusuran penuh. Segmen yang disimpulkan mewakili layanan hilir dan sumber daya di peta jejak.

X-Ray menyediakan Skema JSON untuk dokumen segmen. Anda dapat mengunduh skema di sini: [xray-segmentdocument-schema-v1.0.0](#). Bidang dan objek yang tercantum dalam skema dijelaskan lebih detail pada bagian berikut.

Bagian dari bidang segmen diindeks oleh X-Ray untuk digunakan dengan ekspresi filter. Misalnya, jika Anda mengatur bidang `user` pada sebuah segmen ke pengenal unik, Anda dapat mencari segmen yang terkait dengan pengguna tertentu di konsol X-Ray atau menggunakan API [GetTraceSummaries](#). Untuk informasi selengkapnya, lihat [Menggunakan ekspresi filter](#).

Saat Anda melengkapi aplikasi Anda dengan SDK X-Ray, SDK akan menghasilkan dokumen segmen untuk Anda. Alih-alih mengirim dokumen segmen secara langsung ke X-Ray, SDK mentransmisikannya melalui port UDP lokal kepada [Daemon X-Ray](#). Untuk informasi selengkapnya, lihat [Mengirim dokumen segmen ke daemon X-Ray](#).

Bagian-bagian

- [Bidang segmen](#)
- [Subsegmen](#)
- [Data permintaan HTTP](#)

- [Anotasi](#)
- [Metadata](#)
- [AWS data sumber daya](#)
- [Kesalahan dan pengecualian](#)
- [Kueri SQL](#)

Bidang segmen

Segmen mencatat informasi pelacakan tentang permintaan yang digunakan aplikasi Anda. Minimal, segmen mencatat nama, ID, waktu mulai, penelusuran ID, dan waktu akhir permintaan.

Example Segmen minimal yang lengkap

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

Bidang berikut diperlukan, atau secara kondisional diperlukan, untuk segmen.

Note

Nilai mesti menjadi string (hingga 250 karakter) kecuali dicatat sebaliknya.

Bidang Segment yang Diperlukan

- **name** – Nama logis dari layanan yang menangani permintaan, hingga 200 karakter. Misalnya, nama aplikasi atau nama domain Anda. Nama dapat berisi huruf Unicode, angka, dan spasi, serta simbol-simbol berikut: `_`, `.`, `:`, `/`, `%`, `&`, `#`, `=`, `+`, `\`, `-`, `@`
- **id** – Pengenal 64-bit untuk segmen, unik di antara segmen dalam penelusuran yang sama, dalam 16 digit heksadesimal.
- **trace_id** – Pengenal unik yang menghubungkan semua segmen dan subsegmen yang berasal dari permintaan klien tunggal.

Format ID jejak X-Ray

`trace_id`X-Ray terdiri dari tiga angka yang dipisahkan oleh tanda hubung. Contohnya, `1-58406520-a006649127e371903a2de979`. Hal ini mencakup:

- Nomor versi, yaitu 1.
- Waktu permintaan asli dalam waktu epoch Unix menggunakan 8 digit heksadesimal.

Misalnya, 10:00 AM 1 Desember 2016 PST dalam waktu epoch adalah 1480615200 detik atau 58406520 dalam digit heksadesimal.

- Pengidentifikasi 96-bit yang unik secara global untuk jejak dalam 24 digit heksadesimal.

Note

X-Ray sekarang mendukung jejak IDs yang dibuat menggunakan OpenTelemetry dan kerangka kerja lain yang sesuai dengan spesifikasi [W3C Trace](#) Context. ID jejak W3C harus diformat dalam format X-Ray Trace ID saat mengirim ke X-Ray. Misalnya, ID jejak W3C `4efaaf4d1e8720b39541901950019ee5` harus diformat seperti `1-4efaaf4d-1e8720b39541901950019ee5` saat mengirim ke X-Ray. Jejak X-Ray IDs menyertakan cap waktu permintaan asli dalam waktu epoch Unix, tetapi ini tidak diperlukan saat mengirim jejak IDs W3C dalam format X-Ray.

Keamanan ID Penelusuran

IDs Jejak terlihat di [header respons](#). Hasilkan jejak IDs dengan algoritme acak yang aman untuk memastikan bahwa penyerang tidak dapat menghitung jejak masa depan IDs dan mengirim permintaan IDs ke aplikasi Anda.

- `start_time` – bilangan yang merupakan waktu ketika segmen dibuat, di detik titik mengambang dalam jangka waktu zaman. Misalnya, `1480615200.010` atau `1.480615200010E9`. Gunakan tempat desimal sebanyak yang Anda butuhkan. Resolusi microsecond dianjurkan bila tersedia.
- `end_time` – angka yang merupakan waktu segmen ditutup. Contohnya, `1480615200.090` atau `1.480615200090E9`. Tentukan salah satu dari `end_time` atau `in_progress`.
- `in_progress` – boolean, atur ke `true` alih-alih menentukan sebuah `end_time` untuk mencatat bahwa segmen dimulai, tetapi tidak lengkap. Kirim segmen yang sedang berlangsung saat aplikasi Anda menerima permintaan yang akan memakan waktu lama untuk melayani, untuk melacak tanda

terima permintaan. Ketika respons dikirim, mengirim segmen lengkap untuk menimpa segmen sedang berlangsung. Hanya mengirim satu segmen lengkap, dan satu atau nol segmen dalam proses, per permintaan.

Nama Layanan

Segmen name mesti sesuai dengan nama domain atau nama logis dari layanan yang menghasilkan segmen. Walau bagaimanapun, ini tidak dipaksakan. Aplikasi apa pun yang memiliki izin untuk [PutTraceSegments](#) dapat mengirim segmen dengan nama apa pun.

Bidang berikut opsional untuk segmen.

Bidang Segmen Opsional

- `service` – Objek dengan informasi tentang aplikasi Anda.
 - `version` – String yang mengidentifikasi versi aplikasi Anda yang melayani permintaan.
- `user` – Sebuah string yang mengidentifikasi pengguna yang mengirim permintaan.
- `origin`— Jenis AWS sumber daya yang menjalankan aplikasi Anda.

Nilai yang Didukung

- `AWS::EC2::Instance`— Sebuah EC2 contoh Amazon.
- `AWS::ECS::Container` – Kontainer Amazon ECS.
- `AWS::ElasticBeanstalk::Environment` – Lingkungan Elastic Beanstalk

Ketika beberapa nilai yang berlaku untuk aplikasi Anda, gunakan salah satu yang paling spesifik. Misalnya, lingkungan Multicontainer Docker Elastic Beanstalk menjalankan aplikasi Anda di container Amazon ECS, yang pada gilirannya berjalan pada instance Amazon. EC2 Dalam hal ini Anda akan mengatur asal untuk `AWS::ElasticBeanstalk::Environment` karena lingkungan adalah induk dari dua sumber daya lainnya.

- `parent_id` – ID subsegmen yang Anda tentukan jika permintaan berasal dari aplikasi yang diinstrumentasi. X-Ray SDK menambahkan ID subsegmen induk ke [Header pelacakan](#) untuk panggilan HTTP hilir. Dalam kasus subsegment yang di-nest, subsegmen dapat memiliki segmen atau subsegmen sebagai induknya.
- `http` – objek [http](#) dengan informasi tentang permintaan HTTP asli.

- `aws`— [aws](#) objek dengan informasi tentang AWS sumber daya tempat aplikasi Anda melayani permintaan.
- `error`, `throttle`, `fault`, dan `cause` – bidang [kesalahan](#) yang menunjukkan kesalahan terjadi dan yang mencakup informasi tentang pengecualian yang menyebabkan kesalahan.
- `annotations` – Objek [annotations](#) dengan pasangan nilai kunci yang ingin Anda X-Ray untuk indeks pencarian.
- `metadata` – Objek [metadata](#) dengan data tambahan yang ingin Anda simpan di segmen.
- `subsegments` – susunan objek [subsegment](#).

Subsegmen

Anda dapat membuat subsegmen untuk merekam panggilan Layanan AWS dan sumber daya yang Anda buat dengan AWS SDK, panggilan ke web HTTP internal atau eksternal APIs, atau kueri database SQL. Anda juga dapat membuat subsegment untuk debug atau anotasi blok kode dalam aplikasi Anda. Subsegmen dapat berisi subsegmen lain, sehingga subsegmen kustom yang mencatat metadata tentang panggilan fungsi internal dapat berisi subsegmen dan subsegmen kustom lainnya untuk panggilan hilir.

Sebuah subsegmen mencatat panggilan hilir dari sudut pandang layanan yang menyebutnya. X-Ray menggunakan subsegmen untuk mengidentifikasi layanan hilir yang tidak mengirim segmen dan membuat entri untuk mereka di grafik layanan.

Sebuah subsegmen dapat tertanam dalam dokumen segmen penuh atau dikirim secara independen. Kirim subsegment secara terpisah untuk asynchronously melacak panggilan hilir untuk permintaan berjalan lama, atau untuk menghindari melampaui ukuran dokumen segmen maksimum.

Example Segmen dengan subsegmen tertanam

Subsegmen independen memiliki `type` dari `subsegment` dan `parent_id` yang mengidentifikasi segmen induk.

```
{
  "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
      }
    }
  }
]
}

```

Untuk permintaan yang berjalan lama, Anda dapat mengirim segmen yang sedang berlangsung untuk memberi tahu X-Ray bahwa permintaan telah diterima, lalu mengirim subsegmen secara terpisah untuk melacaknya sebelum menyelesaikan permintaan awal.

Example Segmen yang sedang berlangsung

```

{
  "name" : "example.com",
  "id"   : "70de5b6f19ff9a0b",
  "start_time" : 1.478293361271E9,

```

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

Example Subsegmen independen

Subsegmen independen memiliki type dari subsegment, trace_id, dan parent_id yang mengidentifikasi segmen induk.

```
{
  "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
    }
  }
}
```

Ketika permintaan selesai, tutup segmen dengan mengirim ulanginya bersama end_time. Segmen lengkap menerima segmen yang sedang berlangsung.

Anda juga dapat mengirim subsegment secara terpisah untuk permintaan selesai yang memicu alur kerja asinkron. Misalnya, web API dapat mengembalikan respons OK 200 segera sebelum memulai pekerjaan yang diminta pengguna. Anda dapat mengirim segmen lengkap ke X-Ray segera setelah respons dikirim, diikuti oleh subsegmen untuk pekerjaan yang diselesaikan nanti. Seperti segmen, Anda juga dapat mengirim fragmen subsegmen untuk mencatat bahwa subsegmen telah dimulai, dan kemudian menerima dengan subsegmen penuh setelah panggilan hilir selesai.

Bidang berikut diperlukan, atau kondisional diperlukan, untuk subsegment.

 Note

Nilai adalah string hingga 250 karakter kecuali dicatat sebaliknya.

Bidang Subsegmen yang Wajib

- `id` – Pengenal 64-bit untuk subsegmen, unik di antara segmen di penelusuran yang sama, dalam 16 digit heksadesimal.
- `name` – Nama logis dari subsegmen. Untuk panggilan hilir, nama subsegmen setelah sumber daya atau layanan yang disebut. Untuk subsegment kustom, nama subsegmen setelah kode yang instrumen (misalnya, nama fungsi).
- `start_time` – angka yang merupakan waktu subsegment dibuat, di floating point detik dalam waktu zaman, akurat untuk milidetik. Misalnya, 1480615200.010 atau 1.480615200010E9.
- `end_time` – angka yang merupakan waktu segmen ditutup. Misalnya, 1480615200.090 atau 1.480615200090E9. Tentukan sebuah `end_time` atau `in_progress`.
- `in_progress` – boolean yang diatur ke `true` alih-alih menentukan sebuah `end_time` untuk mencatat bahwa segmen dimulai, tetapi tidak lengkap. Hanya kirim satu subsegmen lengkap, dan satu atau nol subsegmen dalam proses, per permintaan hilir.
- `trace_id` – Penelusuran ID dari segmen induk subsegmen ini. Diperlukan hanya jika mengirim subsegmen secara terpisah.

Format ID jejak X-Ray

`trace_id`X-Ray terdiri dari tiga angka yang dipisahkan oleh tanda hubung. Contohnya, 1-58406520-a006649127e371903a2de979. Hal ini mencakup:

- Nomor versi, yaitu 1.
- Waktu permintaan asli dalam waktu epoch Unix menggunakan 8 digit heksadesimal.

Misalnya, 10:00 AM 1 Desember 2016 PST dalam waktu epoch adalah 1480615200 detik atau 58406520 dalam digit heksadesimal.

- Pengidentifikasi 96-bit yang unik secara global untuk jejak dalam 24 digit heksadesimal.

 Note

X-Ray sekarang mendukung jejak IDs yang dibuat menggunakan OpenTelemetry dan kerangka kerja lain yang sesuai dengan spesifikasi [W3C Trace](#) Context. ID jejak

W3C harus diformat dalam format X-Ray Trace ID saat mengirim ke X-Ray. Misalnya, ID jejak W3C 4efaaf4d1e8720b39541901950019ee5 harus diformat seperti 1-4efaaf4d-1e8720b39541901950019ee5 saat mengirim ke X-Ray. Jejak X-Ray IDs menyertakan cap waktu permintaan asli dalam waktu epoch Unix, tetapi ini tidak diperlukan saat mengirim jejak IDs W3C dalam format X-Ray.

- `parent_id` – Segmen ID dari segmen induk subsegmen ini. Diperlukan hanya jika mengirim subsegmen secara terpisah. Dalam kasus subsegment yang di-nest, subsegmen dapat memiliki segmen atau subsegmen sebagai induknya.
- `type` – subsegment. Diperlukan hanya jika mengirim subsegmen secara terpisah.

Bidang berikut adalah opsional untuk subsegment.

Kolom Subsegmen opsional

- `namespace` – `aws` untuk panggilan AWS SDK; `remote` untuk panggilan hilir lainnya.
- `http` – objek [http](#) dengan informasi tentang panggilan HTTP keluar.
- `aws`— [aws](#) objek dengan informasi tentang AWS sumber daya hilir yang disebut aplikasi Anda.
- `error`, `throttle`, `fault`, dan `cause` – bidang [kesalahan](#) yang menunjukkan kesalahan terjadi dan yang mencakup informasi tentang pengecualian yang menyebabkan kesalahan.
- `annotations` – Objek [annotations](#) dengan pasangan nilai kunci yang ingin Anda X-Ray untuk indeks pencarian.
- `metadata` – Objek [metadata](#) dengan data tambahan yang ingin Anda simpan di segmen.
- `subsegments` – susunan objek [subsegment](#).
- `precursor_ids`— larik subsegmen IDs yang mengidentifikasi subsegmen dengan induk yang sama yang diselesaikan sebelum subsegmen ini.

Data permintaan HTTP

Gunakan blok HTTP untuk mencatat detail permintaan HTTP bahwa aplikasi Anda sediakan (dalam segmen) atau yang aplikasi Anda buat untuk HTTP API hilir (dalam subsegmen). Sebagian besar bidang dalam peta objek ini untuk informasi yang ditemukan dalam permintaan HTTP dan respon.

http

Semua kolom lain bersifat opsional.

- `request` – Informasi tentang permintaan.
 - `method` – Metode permintaan Misalnya, GET.
 - `url` – URL lengkap dari permintaan, disusun dari protokol, hostname, dan path dari permintaan.
 - `user_agent` – String agen pengguna dari klien peminta.
 - `client_ip` – Alamat IP peminta. Dapat diambil dari paket IP Source Address atau, untuk permintaan yang diteruskan, dari header X-Forwarded-For.
 - `x_forwarded_for` – (hanya segmen)boolean menunjukkan bahwa `client_ip` dibaca dari header X-Forwarded-For dan tidak dapat diandalkan karena mungkin telah ditempa.
 - `traced` – (hanya subsegmen)boolean menunjukkan bahwa panggilan hilir adalah untuk layanan lain ditelusuri. Jika bidang ini diatur ke `true`, X-Ray menganggap penelusuran tersebut rusak hingga layanan hilir mengunggah segmen dengan `parent_id` yang cocok dengan `id` dari subsegmen yang berisi blok ini.
- `response` – Informasi tentang respon.
 - `status`— bilangan bulat yang menunjukkan status HTTP dari respons.
 - `content_length`— bilangan bulat yang menunjukkan panjang badan respons dalam byte.

Ketika Anda instrumen panggilan ke api web hilir, mencatat subsegmen dengan informasi tentang permintaan HTTP dan respon. X-Ray menggunakan subsegmen untuk membuat segmen disimpulkan untuk API jarak jauh.

Example Segmen untuk panggilan HTTP yang dilayani oleh aplikasi yang berjalan di 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 Subsegmen untuk panggilan HTTP downstream

```

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

```

Example Segmen yang disimpulkan untuk panggilan HTTP downstream

```

{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,

```



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

Anotasi

Segmen dan subsegmen dapat mencakup objek annotations yang berisi satu atau lebih bidang yang X-Ray indeks untuk digunakan dengan ekspresi filter. Fields dapat memiliki string, angka, atau nilai Boolean (tidak ada objek atau array). Indeks X-Ray hingga 50 anotasi per penelusuran.

Example Segmen untuk panggilan HTTP dengan anotasi

```
{
  "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
  }
}

```

Kunci harus alfanumerik agar bisa bekerja dengan filter. Garis bawah diperbolehkan. Simbol dan spasi lainnya tidak diperbolehkan.

Metadata

Segmen dan subsegmen dapat mencakup objek metadata yang berisi satu atau lebih bidang dengan nilai-nilai dari tipe apa pun, termasuk objek dan array. X-Ray tidak mengindeks metadata, dan nilai dapat berupa ukuran apa pun, selama dokumen segmen tidak melebihi ukuran maksimum (64 kB). Anda dapat melihat metadata dalam dokumen segmen penuh dikembalikan oleh API [BatchGetTraces](#). Kunci bidang (debug dalam contoh berikut) dimulai dengan AWS . dicadangkan untuk digunakan oleh AWS-provided SDKs dan klien.

Example Subsegmen kustom dengan metadata

```

{
  "id": "0e58d2918e9038e8",
  "start_time": 1484789387.502,
  "end_time": 1484789387.534,
  "name": "## UserModel.saveUser",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
  "subsegments": [
    {
      "id": "0f910026178b71eb",

```

```

    "start_time": 1484789387.502,
    "end_time": 1484789387.534,
    "name": "DynamoDB",
    "namespace": "aws",
    "http": {
      "response": {
        "content_length": 58,
        "status": 200
      }
    },
    "aws": {
      "table_name": "scorekeep-user",
      "operation": "UpdateItem",
      "request_id": "3AIENM5J4ELQ3SP0DHKBIRVIC3VV4KQNS05AEMVJF66Q9ASUAAJG",
      "resource_names": [
        "scorekeep-user"
      ]
    }
  }
]
}

```

AWS data sumber daya

Untuk segmen, **aws** objek berisi informasi tentang sumber daya tempat aplikasi Anda berjalan. Beberapa bidang dapat berlaku untuk sumber daya tunggal. Misalnya, aplikasi yang berjalan di lingkungan Docker multicontainer di Elastic Beanstalk dapat memiliki informasi tentang instans Amazon, wadah EC2 Amazon ECS yang berjalan pada instance, dan lingkungan Elastic Beanstalk itu sendiri.

aws (Segmen)

Semua bidang bersifat opsional.

- **account_id**— Jika aplikasi Anda mengirim segmen ke yang berbeda Akun AWS, catat ID akun yang menjalankan aplikasi Anda.
- **cloudwatch_logs**— Array objek yang menggambarkan grup CloudWatch log tunggal.
 - **log_group**— Nama Grup CloudWatch Log.
 - **arn**— Grup CloudWatch Log ARN.
- **ec2**— Informasi tentang EC2 contoh Amazon.

- `instance_id`— ID instance dari EC2 instance.
- `instance_size`— Jenis EC2 contoh.
- `ami_id`— ID Gambar Mesin Amazon.
- `availability_zone` - Availability Zone tempat instans berjalan.
- `ecs` – Informasi tentang kontainer Amazon ECS.
 - `container`— Nama host wadah Anda.
 - `container_id`— ID kontainer lengkap wadah Anda.
 - `container_arn`— ARN dari instance kontainer Anda.
- `eks`— Informasi tentang cluster Amazon EKS.
 - `pod`— Nama host pod EKS Anda.
 - `cluster_name`— Nama cluster EKS.
 - `container_id`— ID kontainer lengkap wadah Anda.
- `elastic_beanstalk` – Informasi tentang lingkungan Elastic Beanstalk. Anda dapat menemukan informasi ini dalam sebuah file bernama `/var/elasticbeanstalk/xray/environment.conf` pada platform Elastic Beanstalk terbaru.
 - `environment_name` – Nama lingkungan.
 - `version_label`— Nama versi aplikasi yang saat ini digunakan untuk instans yang melayani permintaan.
 - `deployment_id`— ID yang menunjukkan ID dari deployment berhasil terakhir ke instans yang melayani permintaan.
- `xray`— Metadata tentang jenis dan versi instrumentasi yang digunakan.
 - `auto_instrumentation`— Boolean menunjukkan apakah instrumentasi otomatis digunakan (misalnya, Agen Java).
 - `sdk_version`— Versi SDK atau agen yang digunakan.
 - `sdk`— Jenis SDK.

Example AWS blokir dengan 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"
    }
  }
}

```

Untuk subsegmen, catat informasi tentang Layanan AWS dan sumber daya yang diakses aplikasi Anda. X-Ray menggunakan informasi ini untuk membuat segmen yang disimpulkan yang mewakili layanan hilir di peta layanan Anda.

aws (Subsegmen)

Semua bidang bersifat opsional.

- **operation**— Nama tindakan API yang dipanggil terhadap sumber daya Layanan AWS atau.
- **account_id**— Jika aplikasi Anda mengakses sumber daya di akun yang berbeda, atau mengirim segmen ke akun yang berbeda, catat ID akun yang memiliki AWS sumber daya yang diakses aplikasi Anda.
- **region** – Jika sumber daya berada di wilayah yang berbeda dari aplikasi Anda, catat wilayahnya. Misalnya, `us-west-2`.
- **request_id** – Pengenal unik untuk permintaan.
- **queue_url** – Untuk operasi pada antrean Amazon SQS, URL antrean ini.
- **table_name** – Untuk operasi pada tabel DynamoDB, nama tabel.

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item

```
{
```

```

{id": "24756640c0d0978a",
"start_time": 1.480305974194E9,
"end_time": 1.4803059742E9,
"name": "DynamoDB",
"namespace": "aws",
"http": {
  "response": {
    "content_length": 60,
    "status": 200
  }
},
"aws": {
  "table_name": "scorekeep-user",
  "operation": "UpdateItem",
  "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
}
}

```

Kesalahan dan pengecualian

Ketika terjadi kesalahan, Anda dapat mencatat detail tentang kesalahan dan pengecualian yang dihasilkan. Catatan kesalahan dalam segmen ketika aplikasi Anda kembali kesalahan ke pengguna, dan di subsegment ketika panggilan hilir kembali kesalahan.

Tipe kesalahan

Tetapkan satu atau beberapa bidang berikut `true` untuk menunjukkan bahwa kesalahan terjadi. Beberapa tipe dapat berlaku jika kesalahan berlipat ganda. Misalnya, `Too Many Requests` kesalahan dari panggilan hilir dapat menyebabkan aplikasi Anda kembali `Internal Server Error`, dalam hal ini ketiga tipe akan berlaku.

- `error` – boolean menunjukkan bahwa kesalahan klien terjadi (kode status respons adalah 4XX klien Error).
- `throttle` – boolean menunjukkan bahwa permintaan telah dicekik (kode status respon 429 Terlalu Banyak Permintaan).
- `fault` – boolean menunjukkan bahwa kesalahan server terjadi (kode status respons adalah 5XX Server Error).

Menunjukkan penyebab kesalahan dengan memasukkan `Penyebab` objek dalam segmen atau subsegment.

cause

Penyebab bisa berupa 16 karakter pengecualian ID atau objek dengan bidang-bidang berikut:

- `working_directory`– Jalur lengkap direktori kerja ketika pengecualian terjadi.
- `paths`– Susun dari path ke pustaka atau modul yang digunakan ketika pengecualian terjadi.
- `exceptions`– Susun dari pengecualian objek.

Sertakan informasi rinci tentang kesalahan dalam satu atau lebih pengecualian objek.

exception

Semua bidang bersifat opsional.

- `id` – Pengenal 64-bit untuk segmen, unik di antara segmen dalam penelusuran yang sama, dalam 16 digit heksadesimal.
- `message`– Pesan pengecualian.
- `type` – Tipe pengecualian.
- `remote`– boolean menunjukkan bahwa pengecualian disebabkan oleh kesalahan yang dikembalikan oleh layanan hilir.
- `truncated` – Integer menunjukkan jumlah frame tumpukan yang dihilangkan dari `stack`.
- `skipped`– Integer menunjukkan jumlah pengecualian yang dilewati antara pengecualian ini dan anaknya, yaitu pengecualian yang ditimbulkannya.
- `cause`– Exception ID dari induk pengecualian ini, yaitu, pengecualian yang menyebabkan pengecualian ini.
- `stack` – Susunan dari objek `stackFrame`.

Jika tersedia, catat informasi tentang tumpukan panggilan di objek `stackFrame`.

stackFrame

Semua kolom lain bersifat opsional.

- `path`– Jalur relatif ke file.
- `line`– Baris di file.
- `label`– Fungsi atau nama metode.

Kueri SQL

Anda dapat membuat subsegmen untuk kueri bahwa aplikasi Anda membuat basis data SQL.

sql

Semua kolom lain bersifat opsional.

- `connection_string` – Untuk SQL Server atau koneksi basis data lain yang tidak menggunakan string koneksi URL, mencatat string koneksi, tidak termasuk kata sandi.
- `url` – Untuk koneksi database yang menggunakan string koneksi URL, catat URL, tidak termasuk password.
- `sanitized_query` – Kueri basis data, dengan setiap pengguna diberikan nilai-nilai dihapus atau diganti dengan placeholder.
- `database_type` – Nama mesin basis data.
- `database_version` – nomor versi mesin basis data yang akan ditingkatkan.
- `driver_version` – Nama dan nomor versi driver mesin basis data yang digunakan aplikasi Anda.
- `user` – Nama pengguna basis data.
- `preparation` – `call` jika kueri menggunakan `PreparedStatement`; `statement` jika kueri menggunakan `PreparedStatement`.

Example Subsegmen dengan Kueri 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 konsep

AWS X-Ray menerima data dari layanan sebagai segmen. X-Ray kemudian mengelompokkan segmen yang memiliki permintaan umum ke penelusuran. X-Ray memproses pelacakan untuk menghasilkan Grafik layanan yang menyediakan representasi visual dari aplikasi Anda.

Konsep

- [Segmen](#)
- [Subsegmen](#)
- [Grafik layanan](#)
- [Pelacakan](#)
- [Pengambilan sampel](#)
- [Header penelusuran](#)
- [Ekspresi filter](#)
- [Grup](#)
- [Anotasi dan metadata](#)
- [Kesalahan, cacat, dan pengecualian](#)

Segmen

Sumber daya komputasi menjalankan logika aplikasi Anda mengirim data tentang pekerjaan mereka sebagai segmen. Segmen menyediakan nama sumber daya, detail tentang permintaan dan pekerjaan yang dilakukan. Misalnya, ketika permintaan HTTP mencapai aplikasi Anda, dapat mencatat data berikut tentang:

- Host – hostname, alias atau alamat IP
- Permintaan – metode, alamat klien, path, agen pengguna
- Tanggapan – status, konten
- Pekerjaan yang dilakukan – waktu mulai dan akhir, subsegmen
- Masalah yang terjadi – [kesalahan, kesalahan dan pengecualian](#), termasuk penangkapan otomatis tumpukan pengecualian.

Segment details: Scorekeep



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

X-Ray SDK mengumpulkan informasi dari header permintaan dan respons, kode dalam aplikasi Anda, dan metadata tentang AWS sumber daya yang dijelankannya. Anda memilih data yang akan dikumpulkan dengan memodifikasi konfigurasi atau kode aplikasi Anda untuk instrumen permintaan masuk, permintaan hilir, dan AWS klien SDK.

Permintaan yang Diteruskan

Jika penyeimbang beban atau perantara lainnya meneruskan permintaan ke aplikasi Anda, X-Ray akan mengambil IP klien dari header `X-Forwarded-For` dalam permintaan bukan dari sumber IP dalam paket IP. IP klien yang dicatat untuk permintaan diteruskan dapat ditempa, sehingga tidak dapat dipercaya.

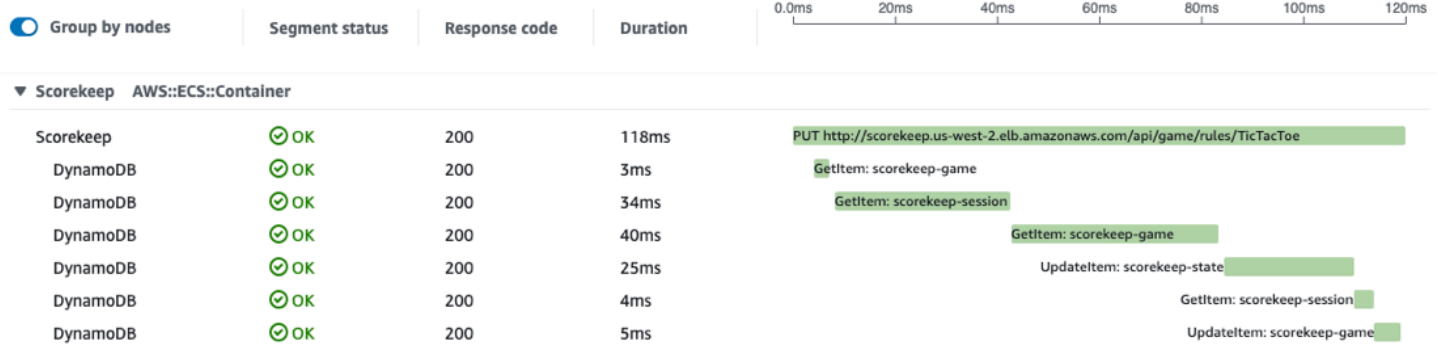
Anda dapat menggunakan X-Ray SDK untuk mencatat informasi tambahan seperti [anotasi dan metadata](#). Untuk detail mengenai struktur dan informasi yang dicatat dalam segmen dan subsegmen, lihat [AWS X-Ray dokumen segmen](#). Dokumen segmen bisa berukuran hingga 64 kB.

Subsegmen

Segmen dapat memecah data tentang pekerjaan yang dilakukan menjadi subsegmen. Subsegmen memberikan informasi waktu yang lebih terperinci dan detail tentang panggilan hilir yang dibuat aplikasi Anda untuk memenuhi permintaan awal. Subsegmen dapat berisi rincian tambahan tentang panggilan ke Layanan AWS, API HTTP eksternal, atau database SQL. Anda bahkan dapat

menentukan subsegmen sewenang-wenang untuk instrumen fungsi tertentu atau baris kode dalam aplikasi Anda.

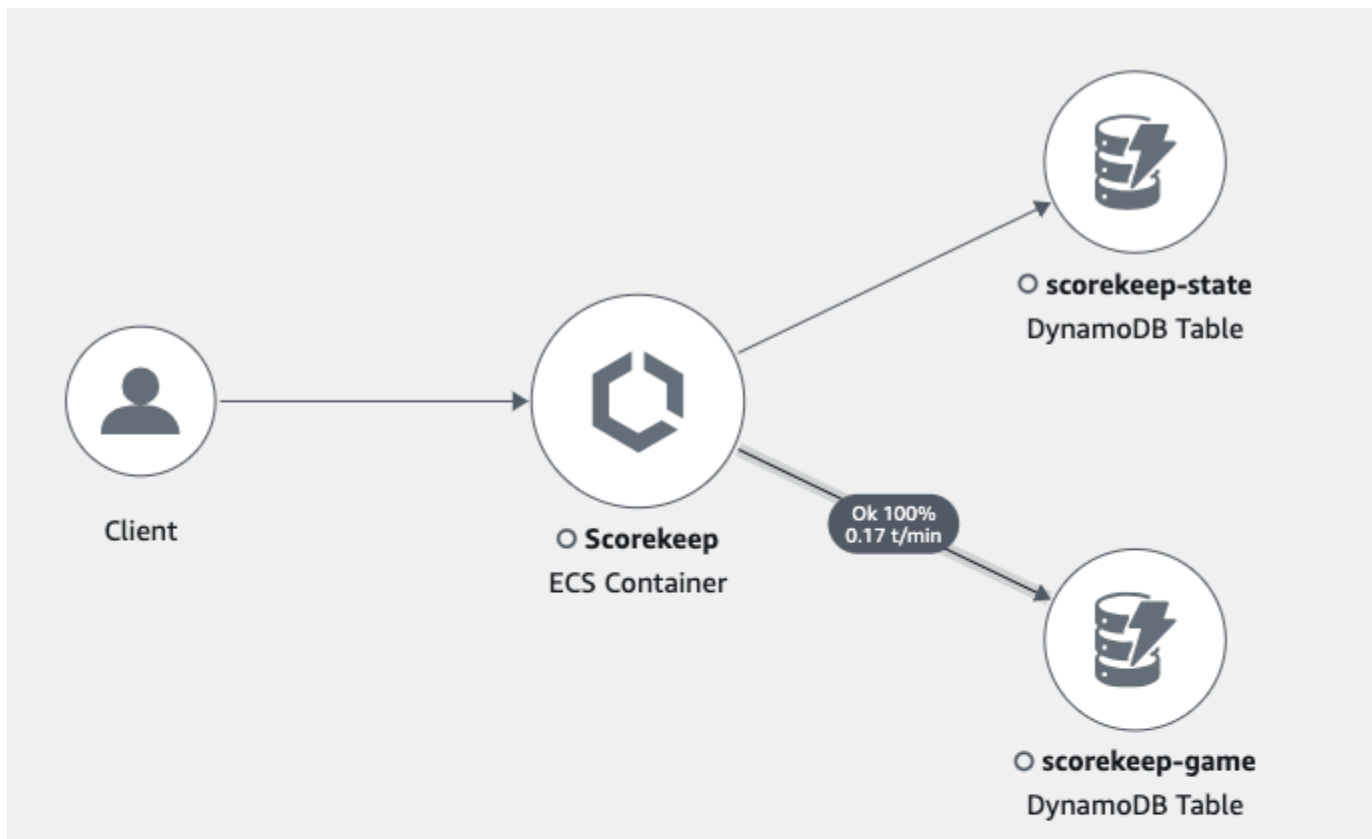
Segments Timeline [Info](#)



Untuk layanan yang tidak mengirim segmennya sendiri, seperti Amazon DynamoDB, X-Ray menggunakan subsegmen untuk menghasilkan segmen yang disimpulkan dan node hilir pada peta jejak. Hal ini memungkinkan Anda melihat semua dependensi hilir, meskipun tidak mendukung penelusuran, atau eksternal.

Subsegmen mewakili tampilan aplikasi Anda dari panggilan hilir sebagai klien. Jika layanan hilir juga diinstrumentasi, segmen yang mengirimkan menggantikan segmen disimpulkan yang dihasilkan dari subsegmen klien hulu. Simpul pada grafik layanan selalu menggunakan informasi dari segmen layanan, jika tersedia, ketika edge antara dua simpul menggunakan subsegmen layanan hulu ini.

Misalnya, saat Anda memanggil DynamoDB dengan klien SDK yang AWS diinstrumentasi, X-Ray SDK merekam subsegmen untuk panggilan tersebut. DynamoDB tidak mengirim segmen, sehingga segmen disimpulkan dalam penelusuran, simpul DynamoDB pada layanan grafik, dan edge antara layanan Anda dan DynamoDB semua berisi informasi dari subsegmen.

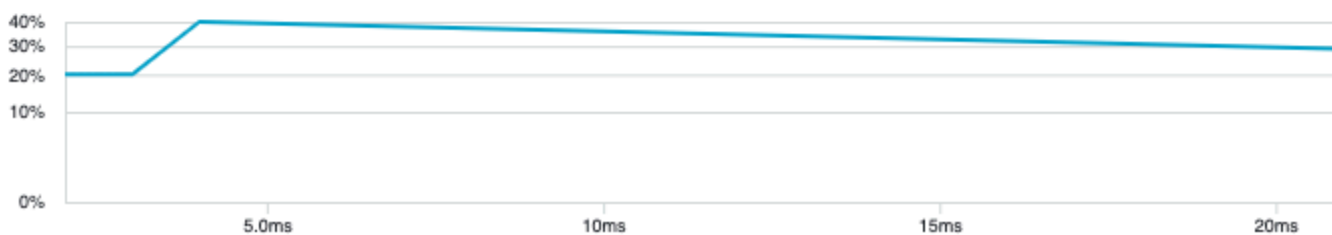


▼ Edge details

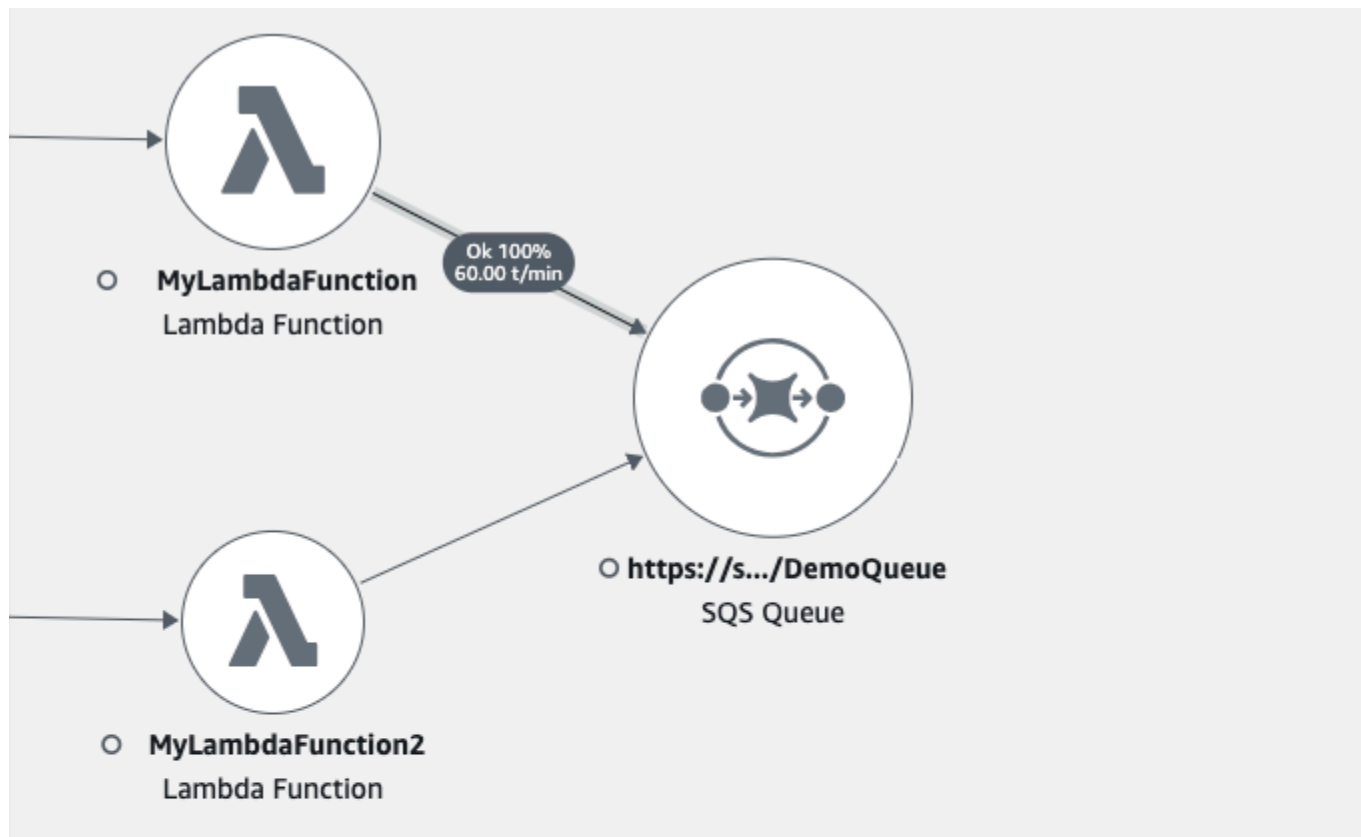
Source: Scorekeep Destination: scorekeep-game

Response time distribution filter

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



Ketika Anda memanggil layanan diinstrumentasi lain dengan aplikasi instrumen, layanan hilir mengirimkan segmen sendiri untuk mencatat pandangan panggilan yang sama bahwa layanan hulu dicatat dalam subsegmen. Dalam grafik layanan, simpul kedua layanan berisi informasi waktu dan kesalahan dari segmen layanan tersebut, sedangkan edge antara layanan berisi informasi dari subsegmen layanan hulu ini.



▼ Edge details

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

Response time distribution filter

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



Kedua sudut pandang bermanfaat, karena layanan hilir mencatat dengan tepat ketika mulai dan berakhir bekerja pada permintaan, dan layanan hulu mencatat latensi trip, termasuk waktu yang dihabiskan permintaan bepergian antara dua layanan.

Grafik layanan

X-Ray menggunakan data yang dikirimkan aplikasi Anda untuk menghasilkan Grafik layanan. Setiap AWS sumber daya yang mengirimkan data ke X-Ray muncul sebagai layanan dalam grafik. Edge menghubungkan layanan yang bekerja sama untuk melayani permintaan. Edge menghubungkan klien pada aplikasi Anda, dan aplikasi Anda untuk layanan hilir dan sumber daya yang menggunakan.

Nama Layanan

Segmen name mesti sesuai dengan nama domain atau nama logis dari layanan yang menghasilkan segmen. Walau bagaimanapun, ini tidak dipaksakan. Aplikasi apa pun yang memiliki izin untuk [PutTraceSegments](#) dapat mengirim segmen dengan nama apa pun.

Sebuah grafik layanan adalah dokumen JSON yang berisi informasi tentang layanan dan sumber daya yang membentuk aplikasi Anda. Konsol X-Ray menggunakan grafik layanan untuk menghasilkan visualisasi atau Peta layanan.



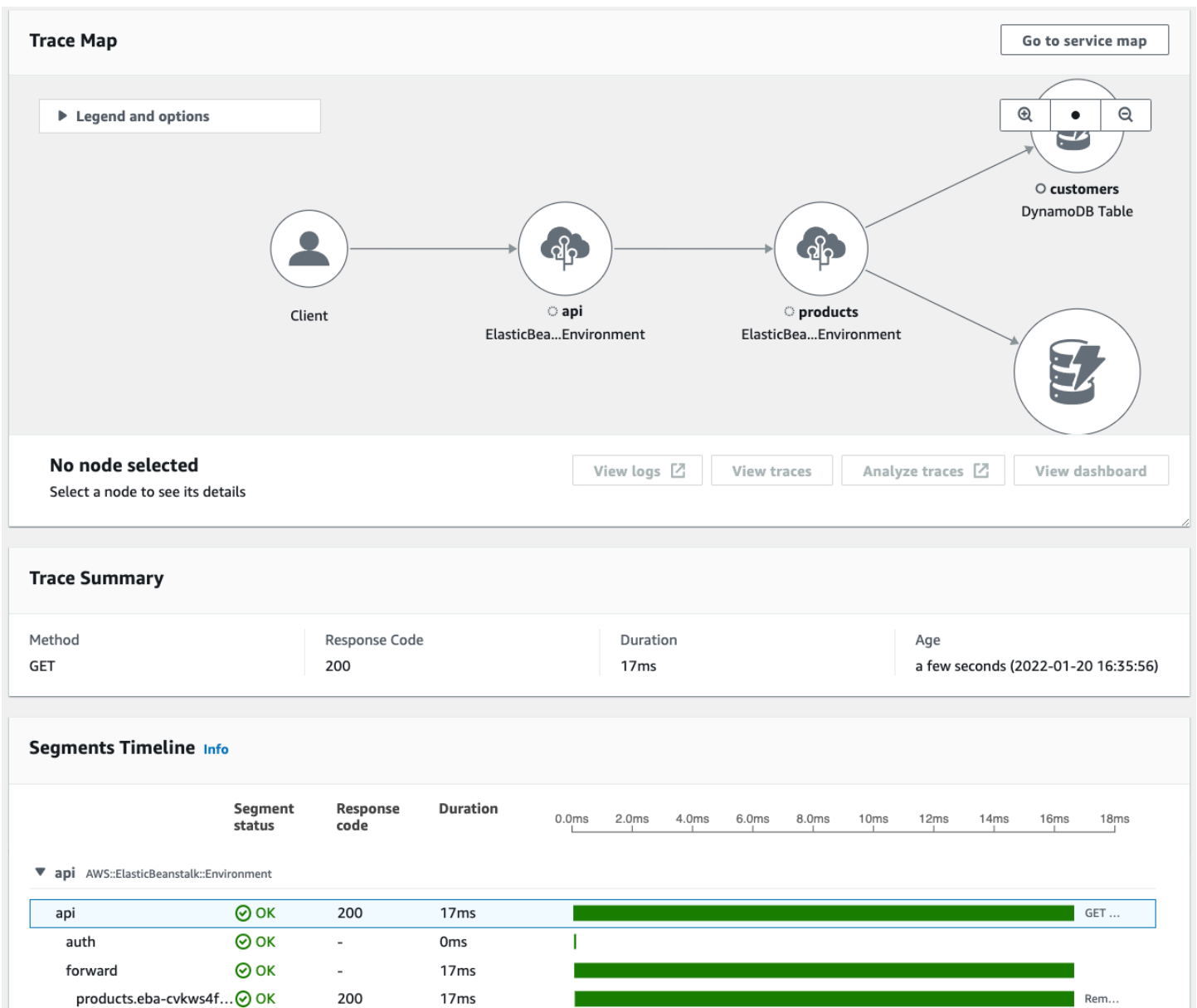
Untuk aplikasi terdistribusi, X-Ray menggabungkan simpul dari semua layanan yang memproses permintaan dengan ID penelusuran yang sama ke dalam satu layanan grafik. Layanan pertama yang temuan permintaannya menambahkan [Header penelusuran](#) yang disebarkan antara front end dan layanan yang disebut.

Misalnya, [Skorrekeep](#) menjalankan API web yang memanggil microservice (sebuah fungsi AWS Lambda) untuk menghasilkan nama acak dengan menggunakan pustaka Node.js. X-Ray SDK for Java menghasilkan ID penelusuran dan termasuk dalam panggilan ke Lambda. Lambda mengirimkan penelusuran data dan melewati penelusuran ID ke fungsi. SDK X-Ray untuk Node.js juga menggunakan ID penelusuran untuk mengirim data. Akibatnya, node untuk API, layanan Lambda, dan fungsi Lambda semuanya muncul sebagai node terpisah, tetapi terhubung, pada peta jejak.

Data grafik layanan dipertahankan selama 30 hari.

Pelacakan

ID Penelusuran menelusuri jalur permintaan melalui aplikasi Anda. Penelusuran mengumpulkan semua segmen yang dihasilkan oleh satu permintaan. Permintaan itu biasanya permintaan HTTP GET atau POST yang berjalan melalui penyeimbang beban, menekan kode aplikasi Anda, dan menghasilkan panggilan hilir ke AWS layanan lain atau web eksternal. APIs Layanan pertama didukung bahwa permintaan HTTP berinteraksi dengan menambahkan penelusuran ID header untuk permintaan, dan menyebarkan hilir untuk melacak latensi, disposisi, dan data permintaan lainnya.



Lihat [AWS X-Ray harga](#) untuk informasi tentang bagaimana jejak X-Ray ditagih. Data jejak disimpan selama 30 hari.

Pengambilan sampel

SDK X-Ray menerapkan algoritme sampling untuk memastikan pelacakan efisien, seraya tetap memberikan sampel representatif dari permintaan yang dilayani oleh aplikasi Anda. Secara default, SDK X-Ray mencatat permintaan pertama setiap detik, dan lima persen permintaan tambahan apa pun.

Untuk menghindari timbulnya biaya layanan ketika Anda memulai, hitungan sampling default adalah konservatif. Anda dapat mengonfigurasi X-Ray untuk mengubah aturan pengambilan sampel default dan mengonfigurasi aturan tambahan yang menerapkan pengambilan sampel berdasarkan properti layanan atau permintaan.

Misalnya, Anda mungkin ingin menonaktifkan sampling dan melacak semua permintaan panggilan yang mengubah status atau menangani pengguna atau transaksi. Untuk panggilan baca-saja dengan volume tinggi, seperti polling latar belakang, pemeriksaan kondisi, atau pemeliharaan koneksi, Anda dapat mengalami kecepatan rendah dan masih mendapatkan data yang cukup untuk melihat masalah yang muncul.

Untuk informasi selengkapnya, lihat [Mengonfigurasi aturan pengambilan sampel](#).

Header penelusuran

Semua permintaan ditelusuri, hingga minimum yang dapat dikonfigurasi. Setelah mencapai tahap minimum itu, persentase permintaan ditelusuri untuk menghindari biaya yang tidak perlu. Keputusan pengambilan sampel dan penelusuran ID ditambahkan ke permintaan HTTP pada Penelusuran header bernama `X-Amzn-Trace-Id`. X-Ray-integrated Layanan pertama yang terkena permintaan menambahkan header penelusuran, yang dibaca oleh X-Ray SDK dan disertakan dalam respons.

Example Header pelacakan dengan akar pelacakan ID dan pengambilan sampel keputusan

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

Keamanan Header Pelacakan

Header penelusuran dapat berasal dari X-Ray SDK Layanan AWS, atau permintaan klien. Aplikasi Anda dapat menghapus `X-Amzn-Trace-Id` dari permintaan yang masuk untuk menghindari masalah yang disebabkan oleh pengguna yang menambahkan keputusan penelusuran IDs atau pengambilan sampel ke permintaan mereka.

Header pelacakan juga dapat berisi ID segmen induk jika permintaan berasal dari aplikasi terinstrumentasi. Misalnya, jika aplikasi Anda memanggil API web HTTP hilir dengan klien HTTP yang diinstrumentasi, SDK X-Ray menambahkan ID segmen untuk permintaan asli ke header pelacakan

permintaan hilir. Aplikasi instrumen yang melayani permintaan hilir dapat mencatat ID segmen induk untuk menghubungkan dua permintaan.

Example Header pelacakan dengan akar penelusuran ID dan pengambilan sampel keputusan

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

Lineage dapat ditambahkan ke header jejak oleh Lambda dan Layanan AWS lainnya sebagai bagian dari mekanisme pemrosesan mereka, dan tidak boleh digunakan secara langsung.

Example Menelusuri header dengan Lineage

```
X-Amzn-Trace-Id: Root=1-5759e988-
bd862e3fe1be46a994272793;Parent=53995c3f42cd8ad8;Sampled=1;Lineage=a87bd80c:1|
68fd508a:5|c512fbe3:2
```

Ekspresi filter

Bahkan dengan sampling, aplikasi yang kompleks menghasilkan banyak data. AWS X-Ray Konsol menyediakan easy-to-navigate tampilan grafik layanan. Ini menunjukkan kondisi dan performa informasi yang membantu Anda mengidentifikasi masalah dan peluang untuk optimasi dalam aplikasi Anda. Untuk penelusuran lanjutan, Anda dapat menelusuri pelacakan permintaan individual, atau menggunakan ekspresi filter untuk menemukan pelacakan yang terkait dengan jalur atau pengguna tertentu.

Traces Info 5m 15m 30m

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

Filter by X-Ray group

Run query 5 traces retrieved

► Query refiners

Traces (5)
This table shows the most recent traces with an average response time of 0.16s. It shows as many as 1000 traces.

ID	Trace status	Timestamp	Response code	Response Time	Duration	HTTP Method
...561513004630e58c75c992ed	OK	3.4min (2023-08-16 17:39:20)	200	0.104s	0.104s	POST
...2e83714b7daac593167d2e73	OK	3.4min (2023-08-16 17:39:19)	200	0.07s	0.07s	POST
...54740787431329383155f154	OK	3.4min (2023-08-16 17:39:18)	200	0.1s	0.1s	POST

Grup

Memperluas ekspresi filter, X-Ray juga mendukung fitur grup. Menggunakan ekspresi filter, Anda dapat menentukan kriteria yang digunakan untuk menerima penelusuran ke dalam grup.

Anda dapat memanggil grup berdasarkan nama atau dengan Amazon Resource Name (ARN) untuk menghasilkan grafik layanannya sendiri, ringkasan jejak, dan metrik Amazon. CloudWatch Setelah grup dibuat, pelacakan masuk diperiksa terhadap ekspresi filter grup seperti yang disimpan dalam layanan X-Ray layanan . Metrik untuk jumlah jejak yang cocok dengan setiap kriteria dipublikasikan untuk CloudWatch setiap menit.

Memperbarui ekspresi filter grup tidak mengubah data yang sudah dicatat. Pembaruan hanya berlaku untuk pelacakan berikutnya. Hal ini dapat mengakibatkan grafik gabungan dari ekspresi baru dan lama. Untuk menghindari hal tersebut, hapus grup saat ini dan buat yang baru.

Note

Grup ditagih berdasarkan jumlah pelacakan yang diambil dan cocok dengan ekspresi filter. Untuk informasi selengkapnya, lihat [Harga AWS X-Ray](#).

Untuk informasi selengkapnya tentang grup, lihat [Mengkonfigurasi grup](#).

Anotasi dan metadata

Saat Anda menginstruksikan aplikasi Anda, X-Ray SDK mencatat informasi tentang permintaan masuk dan keluar, AWS sumber daya yang digunakan, dan aplikasi itu sendiri. Anda dapat menambahkan informasi lain ke dokumen segmen sebagai anotasi dan metadata. Anotasi dan metadata dikumpulkan pada tingkat penelusuran, dan dapat ditambahkan ke setiap segmen atau subsegmen.

Anotasi adalah pasangan kunci-nilai sederhana yang diindeks untuk digunakan dengan [ekspresi filter](#). Gunakan anotasi untuk mencatat data yang ingin Anda gunakan untuk mengelompokkan penelusuran pada konsol, atau saat memanggil API [GetTraceSummaries](#).

Indeks X-Ray hingga 50 anotasi per penelusuran.

Metadata adalah pasangan kunci-nilai dengan nilai-nilai dari tipe apa pun, termasuk objek dan daftar, tapi itu tidak diindeks. Gunakan metadata untuk mencatat data yang ingin Anda simpan di penelusuran tetapi tidak perlu digunakan untuk mencari pelacakan.

Anda dapat melihat anotasi dan metadata di jendela detail segmen atau subsegmen, di dalam halaman [Trace details](#) di konsol. CloudWatch

▼ DynamoDB AWS::DynamoDB::Table					
DynamoDB	✓ OK	200	9ms	GetItem: scorekeep-session	
DynamoDB	✓ OK	200	10ms	UpdateItem: scorekeep-game	
DynamoDB	✓ OK	200	46ms	GetItem: scorekeep-session	
DynamoDB	✓ OK	200	39ms		

Segment details: DynamoDB

Overview	Resources	Annotations	Metadata	Exceptions	SQL

Kesalahan, cacat, dan pengecualian

X-Ray melacak kesalahan yang terjadi dalam kode aplikasi Anda, dan kesalahan yang dikembalikan oleh layanan hilir. Kesalahan dikategorikan sebagai berikut.

- **Error** – Kesalahan klien (400 seri kesalahan)
- **Fault** – Server kesalahan (500 seri kesalahan)
- **Throttle** – Kesalahan throttling (429 Terlalu Banyak Permintaan)

Ketika pengecualian terjadi saat aplikasi Anda melayani permintaan instrumented, SDK X-Ray mencatat detail tentang pengecualian, termasuk pelacakan tumpukan, jika tersedia. Anda dapat melihat pengecualian di bawah [Detail Segment](#) di konsol X-Ray.

Keamanan di AWS X-Ray

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model tanggung jawab bersama](#) menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari [program kepatuhan AWS](#). Untuk mempelajari tentang program kepatuhan yang berlaku untuk X-Ray, lihat [Layanan AWS di Cakupan berdasarkan Program Kepatuhan](#).
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini akan membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan X-Ray. Topik berikut menunjukkan cara mengonfigurasi X-Ray untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga akan belajar cara menggunakan Layanan AWS yang lain yang dapat membantu Anda memantau dan mengamankan sumber daya X-Ray Anda.

Topik

- [Perlindungan data di AWS X-Ray](#)
- [Identitas dan manajemen akses untuk AWS X-Ray](#)
- [Validasi kepatuhan untuk AWS X-Ray](#)
- [Ketahanan di AWS X-Ray](#)
- [Keamanan infrastruktur di AWS X-Ray](#)

Perlindungan data di AWS X-Ray

AWS X-Ray selalu mengenkripsi jejak dan data terkait saat istirahat. Saat Anda perlu mengaudit dan menonaktifkan kunci enkripsi untuk kepatuhan atau persyaratan internal, Anda dapat mengonfigurasi

X-Ray untuk menggunakan kunci AWS Key Management Service (AWS KMS) untuk mengenkripsi data.

X-Ray memberikan Kunci yang dikelola AWS nama `aws/xray`. Gunakan kunci ini ketika Anda hanya ingin [mengaudit penggunaan kunci di AWS CloudTrail](#) dan tidak perlu mengelola kunci itu sendiri. Saat Anda perlu mengelola akses ke kunci atau mengonfigurasi rotasi kunci, Anda dapat [membuat kunci yang dikelola pelanggan](#).

Saat Anda mengubah pengaturan enkripsi, X-Ray menghabiskan beberapa waktu untuk membuat dan menyebarkan kunci data. Saat kunci baru sedang diproses, X-Ray dapat mengenkripsi data dengan kombinasi pengaturan baru dan lama. Data yang ada tidak dienkripsi ulang saat Anda mengubah pengaturan enkripsi.

 Note

AWS KMS mengisi daya saat X-Ray menggunakan kunci KMS untuk mengenkripsi atau mendekripsi data jejak.

- Enkripsi default – Gratis.
- Kunci yang dikelola AWS— Bayar untuk penggunaan kunci.
- kunci yang dikelola pelanggan - Bayar untuk penyimpanan dan penggunaan kunci.

Lihat [AWS Key Management Service Harga](#) untuk detailnya.

 Note

Pemberitahuan wawasan X-Ray mengirimkan peristiwa ke Amazon EventBridge, yang saat ini tidak mendukung kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat [Perlindungan Data di Amazon EventBridge](#).

Anda harus memiliki akses tingkat pengguna ke kunci yang dikelola pelanggan untuk mengonfigurasi X-Ray untuk menggunakannya, dan kemudian melihat jejak terenkripsi. Untuk informasi selengkapnya, lihat [Izin pengguna untuk enkripsi](#).

CloudWatch console

Untuk mengkonfigurasi X-Ray untuk menggunakan kunci KMS untuk enkripsi menggunakan konsol CloudWatch

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Pilih Pengaturan di panel navigasi kiri.
3. Pilih Lihat pengaturan di bawah Enkripsi dalam bagian jejak X-Ray.
4. Pilih Edit di bagian konfigurasi Enkripsi.
5. Pilih Gunakan tombol KMS.
6. Pilih kunci dari menu dropdown:
 - aws/xray - Gunakan. Kunci yang dikelola AWS
 - alias kunci — Gunakan kunci yang dikelola pelanggan di akun Anda.
 - Masukkan kunci ARN secara manual — Gunakan kunci yang dikelola pelanggan di akun yang berbeda. Masukkan Amazon Resource Name (ARN) kunci yang lengkap di bidang yang muncul.
7. Pilih Perbarui enkripsi.

X-Ray console

Untuk mengkonfigurasi X-Ray untuk menggunakan kunci KMS untuk enkripsi menggunakan konsol X-Ray

1. Buka [konsol X-Ray](#).
2. Pilih Enkripsi.
3. Pilih Gunakan tombol KMS.
4. Pilih kunci dari menu dropdown:
 - aws/xray - Gunakan. Kunci yang dikelola AWS
 - alias kunci — Gunakan kunci yang dikelola pelanggan di akun Anda.
 - Masukkan kunci ARN secara manual — Gunakan kunci yang dikelola pelanggan di akun yang berbeda. Masukkan Amazon Resource Name (ARN) kunci yang lengkap di bidang yang muncul.

5. Pilih Terapkan.

Note

X-Ray tidak mendukung tombol KMS asimetris.

Jika X-Ray tidak dapat mengakses kunci enkripsi Anda, penyimpanan data akan berhenti. Hal ini dapat terjadi jika pengguna Anda kehilangan akses ke kunci KMS, atau jika Anda menonaktifkan kunci yang sedang digunakan. Saat hal ini terjadi, X-Ray akan menampilkan notifikasi di bilah navigasi.

Untuk mengonfigurasi pengaturan enkripsi dengan X-Ray API, lihat [Mengkonfigurasi pengaturan sampling, grup, dan enkripsi dengan API AWS X-Ray](#).

Identitas dan manajemen akses untuk AWS X-Ray

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengendalikan siapa yang dapat terautentikasi (masuk) dan berwenang (memiliki izin) untuk menggunakan sumber daya X-Ray. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Bagaimana AWS X-Ray bekerja dengan IAM](#)
- [AWS X-Ray contoh kebijakan berbasis identitas](#)
- [Memecahkan masalah AWS X-Ray identitas dan akses](#)

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di X-Ray.

Pengguna layanan – Jika Anda menggunakan layanan X-Ray untuk melakukan tugas Anda, administrator Anda akan memberikan kredensial dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur X-Ray untuk melakukan pekerjaan, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda untuk meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di X-Ray, lihat [Memecahkan masalah AWS X-Ray identitas dan akses](#).

Administrator layanan – Jika Anda bertanggung jawab atas sumber daya X-Ray di perusahaan Anda, Anda mungkin memiliki akses penuh ke X-Ray. Tugas Anda adalah menentukan fitur dan sumber daya X-Ray mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari selengkapnya tentang cara perusahaan Anda dapat menggunakan IAM dengan X-Ray, lihat [Bagaimana AWS X-Ray bekerja dengan IAM](#).

Administrator IAM – Jika Anda adalah administrator IAM, Anda mungkin ingin belajar dengan lebih detail tentang cara Anda menulis kebijakan untuk mengelola akses ke X-Ray. Untuk melihat contoh kebijakan berbasis identitas X-Ray yang dapat Anda gunakan di IAM, lihat [AWS X-Ray contoh kebijakan berbasis identitas](#).

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS](#).

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara

kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Guna mengetahui informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [AWS Signature Version 4 untuk permintaan API](#) dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Autentikasi multi-faktor AWS di IAM](#) dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat meminta kelompok untuk menyebutkan IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat [Kasus penggunaan untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Untuk mengambil peran IAM sementara AWS Management Console, Anda dapat [beralih dari pengguna ke peran IAM \(konsol\)](#). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat [Metode untuk mengambil peran](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Buat peran untuk penyedia identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM](#) dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin

melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.

- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).
- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS](#) dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada EC2 instance dan membuat AWS CLI atau AWS permintaan API. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di Panduan Pengguna](#) IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan

diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat [Pilih antara kebijakan yang dikelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya,

administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus [menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLs mirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACLs. Untuk mempelajari selengkapnya ACLs, lihat [Ringkasan daftar kontrol akses \(ACL\)](#) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS

Untuk informasi selengkapnya tentang Organizations dan SCPs, lihat [Kebijakan kontrol layanan](#) di Panduan AWS Organizations Pengguna.

- Kebijakan kontrol sumber daya (RCPs) — RCPs adalah kebijakan JSON yang dapat Anda gunakan untuk menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda tanpa memperbarui kebijakan IAM yang dilampirkan ke setiap sumber daya yang Anda miliki. RCP membatasi izin untuk sumber daya di akun anggota dan dapat memengaruhi izin efektif untuk identitas, termasuk Pengguna root akun AWS, terlepas dari apakah itu milik organisasi Anda. Untuk informasi selengkapnya tentang Organizations dan RCPs, termasuk daftar dukungan Layanan AWS tersebut RCPs, lihat [Kebijakan kontrol sumber daya \(RCPs\)](#) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

Bagaimana AWS X-Ray bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke X-Ray, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan dengan X-Ray. Untuk mendapatkan tampilan tingkat tinggi tentang cara Layanan AWS kerja X-Ray dan lainnya dengan IAM, lihat Layanan AWS That [Work with IAM di Panduan](#) Pengguna IAM.

Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk memberikan izin X-Ray kepada pengguna dan menghitung sumber daya di akun Anda. IAM mengontrol akses ke layanan X-Ray di tingkat API untuk menerapkan izin secara seragam, terlepas dari klien mana (konsol, AWS SDK,) yang digunakan pengguna Anda. AWS CLI

Untuk [menggunakan konsol X-Ray](#) untuk melihat peta dan segmen jejak, Anda hanya perlu izin baca. Untuk mengaktifkan akses konsol, tambahkan `AWSXrayReadOnlyAccess` [kebijakan terkelola](#) untuk pengguna IAM Anda.

Untuk [pengembangan dan pengujian lokal](#), buat peran IAM dengan izin baca dan tulis. [Asumsikan peran](#) dan simpan kredensi sementara untuk peran tersebut. Anda dapat menggunakan kredensial ini dengan daemon X-Ray, the AWS CLI, dan SDK. AWS Lihat [menggunakan kredensial keamanan sementara dengan AWS CLI](#) untuk informasi selengkapnya.

Untuk [menerapkan aplikasi yang diinstrumentasi AWS](#), buat peran IAM dengan izin tulis dan tetapkan ke sumber daya yang menjalankan aplikasi Anda. `AWSXRayDaemonWriteAccess` termasuk izin untuk mengunggah jejak, dan beberapa izin baca juga untuk mendukung penggunaan aturan [pengambilan sampel](#).

Kebijakan baca dan tulis tidak termasuk izin untuk mengonfigurasi [pengaturan kunci enkripsi](#) dan aturan pengambilan sampel. Gunakan `AWSXrayFullAccess` untuk mengakses setelan ini, atau menambahkan [konfigurasi APIs](#) dalam kebijakan khusus. Untuk enkripsi dan dekripsi dengan kunci yang dikelola pelanggan yang Anda buat, Anda juga perlu [izin untuk menggunakan kunci](#).

Topik

- [Kebijakan berbasis identitas X-Ray](#)
- [Kebijakan berbasis sumber daya X-Ray](#)
- [Otorisasi berdasarkan tanda X-Ray](#)
- [Menjalankan aplikasi Anda secara lokal](#)
- [Menjalankan aplikasi Anda di AWS](#)
- [Izin pengguna untuk enkripsi](#)

Kebijakan berbasis identitas X-Ray

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan apakah tindakan dan sumber daya diizinkan atau ditolak, serta persyaratan terkait diizinkan atau ditolaknya tindakan tersebut. X-Ray mendukung tindakan, sumber daya, dan kunci syarat tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat [Referensi Elemen Kebijakan JSON IAM](#) dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Tindakan kebijakan di X-Ray menggunakan prefiks berikut sebelum tindakan: `xray:`. Misalnya, untuk memberikan izin kepada seseorang untuk mengambil detail sumber daya grup dengan operasi API `GetGroup` X-Ray, Anda mencantumkan tindakan `xray:GetGroup` dalam kebijakan mereka. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. X-Ray menentukan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

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

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Get`, sertakan tindakan berikut:

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

Untuk melihat daftar tindakan X-Ray, lihat [Tindakan yang Ditentukan oleh AWS X-Ray](#) dalam Panduan Pengguna IAM.

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan [Amazon Resource Name \(ARN\)](#). Anda dapat melakukan ini untuk

tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin tingkat sumber daya, misalnya operasi pencantuman, gunakan karakter wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku bagi semua sumber daya.

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

Anda dapat mengontrol akses ke sumber daya dengan menggunakan kebijakan IAM. Untuk tindakan yang mendukung sumber daya tingkat izin, Anda menggunakan Amazon Resource Name (ARN) untuk mengidentifikasi sumber daya yang diberlakukan oleh kebijakan tersebut.

Semua tindakan X-Ray dapat digunakan dalam kebijakan IAM untuk memberikan atau menolak izin pengguna untuk menggunakan tindakan tersebut. Namun, tidak semua [tindakan X-Ray](#) mendukung izin tingkat sumber daya, yang memungkinkan Anda untuk menentukan sumber daya tempat tindakan dapat dilakukan.

Untuk tindakan yang tidak mendukung izin tingkat sumber daya, Anda harus menggunakan "*" sebagai sumber daya.

Tindakan X-Ray berikut mendukung izin tingkat sumber daya:

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

Berikut ini adalah contoh kebijakan izin berbasis identitas untuk tindakan CreateGroup. Contoh ini menunjukkan penggunaan ARN yang berkaitan dengan nama Grup local-users dengan ID unik sebagai wildcard. ID unik terbuat ketika grup dibuat, sehingga tidak dapat diprediksi di kebijakan sebelumnya. Saat menggunakan GetGroup, UpdateGroup, atau DeleteGroup, Anda dapat menentukan ini sebagai wildcard atau ARN yang tepat, termasuk ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:CreateGroup"
      ],
      "Resource": [
        "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
      ]
    }
  ]
}
```

Berikut ini adalah contoh kebijakan izin berbasis identitas untuk tindakan `CreateSamplingRule`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:CreateSamplingRule"
      ],
      "Resource": [
        "arn:aws:xray:eu-west-1:123456789012:sampling-rule/base-scorekeep"
      ]
    }
  ]
}
```

Note

ARN dari aturan pengambilan sampel ditentukan berdasarkan namanya. Tidak seperti grup ARNs, aturan pengambilan sampel tidak memiliki ID yang dihasilkan secara unik.

Untuk melihat daftar jenis sumber daya X-Ray dan jenisnya ARNs, lihat [Sumber Daya yang Ditentukan oleh AWS X-Ray](#) dalam Panduan Pengguna IAM. Untuk mempelajari tindakan mana yang dapat menentukan ARN setiap sumber daya, lihat [Tindakan yang Ditentukan oleh AWS X-Ray](#).

Kunci syarat

X-Ray tidak menyediakan kunci syarat khusus layanan, tetapi mendukung penggunaan beberapa kunci syarat global. Untuk melihat semua kunci kondisi AWS global, lihat [Kunci Konteks Kondisi AWS Global](#) di Panduan Pengguna IAM.

Contoh

Untuk melihat contoh kebijakan berbasis identitas X-Ray, lihat [AWS X-Ray contoh kebijakan berbasis identitas](#).

Kebijakan berbasis sumber daya X-Ray

[X-Ray mendukung kebijakan berbasis sumber daya untuk Layanan AWS integrasi saat ini dan masa depan, seperti penelusuran aktif Amazon SNS](#). Kebijakan berbasis sumber daya X-Ray dapat diperbarui oleh AWS Management Console s lain, atau melalui SDK AWS atau CLI. Misalnya, konsol Amazon SNS mencoba mengonfigurasi kebijakan berbasis sumber daya secara otomatis untuk mengirim jejak ke X-Ray. Dokumen kebijakan berikut memberikan contoh konfigurasi kebijakan berbasis sumber daya X-Ray secara manual.

Example Contoh kebijakan berbasis sumber daya X-Ray untuk penelusuran aktif Amazon SNS

Contoh dokumen kebijakan ini menentukan izin yang diperlukan Amazon SNS untuk mengirim data jejak ke 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"
    }
  }
}
]
}

```

Gunakan CLI untuk membuat kebijakan berbasis sumber daya yang memberikan izin Amazon SNS untuk mengirim data jejak ke 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" } } } ] }'

```

Untuk menggunakan contoh ini, ganti *partition*, *region*, *account-id*, dan *topic-name* dengan AWS partisi tertentu, wilayah, ID akun, dan nama topik Amazon SNS. Untuk memberikan izin kepada semua topik Amazon SNS untuk mengirim data jejak ke X-Ray, ganti nama topik dengan `*`.

Otorisasi berdasarkan tanda X-Ray

Anda dapat melampirkan tanda ke grup X-Ray atau aturan pengambilan sampel, atau meneruskan tanda dalam permintaan ke X-Ray. Untuk mengendalikan akses berdasarkan tanda, Anda dapat memberikan informasi tentang tanda di kebijakan [elemen syarat](#) menggunakan kunci syarat `xray:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang penandaan sumber daya X-Ray, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat [Mengelola akses ke grup X-Ray dan aturan pengambilan sampel berdasarkan tanda](#).

Menjalankan aplikasi Anda secara lokal

Aplikasi Anda mengirimkan data pelacakan ke daemon X-Ray. Daemon mem-buffer dokumen segmen dan mengunggahnya ke layanan X-Ray dalam batch. Daemon memerlukan izin tulis untuk mengunggah data pelacakan dan telemetri ke layanan X-Ray.

Saat Anda [menjalankan daemon secara lokal](#), buat peran IAM, ambil [peran dan simpan kredensi](#) sementara dalam variabel lingkungan, atau dalam file bernama `credentials` dalam folder bernama di folder pengguna Anda. . aws Lihat [menggunakan kredensial keamanan sementara dengan AWS CLI](#) untuk informasi selengkapnya.

Example `~/.aws/credentials`

```
[default]
aws_access_key_id={access key ID}
aws_secret_access_key={access key}
aws_session_token={AWS session token}
```

Jika Anda sudah mengonfigurasi kredensial untuk digunakan dengan AWS SDK atau AWS CLI, daemon dapat menggunakannya. Jika beberapa profil tersedia, daemon menggunakan profil default.

Menjalankan aplikasi Anda di AWS

Saat menjalankan aplikasi AWS, gunakan peran untuk memberikan izin ke EC2 instance Amazon atau fungsi Lambda yang menjalankan daemon.

- Amazon Elastic Compute Cloud (Amazon EC2) — Buat peran IAM dan lampirkan ke EC2 instance sebagai profil [instance](#).
- Amazon Elastic Container Service (Amazon ECS) – Buat IAM role dan lampirkan ke instans kontainer sebagai [IAM role instans kontainer](#).
- AWS Elastic Beanstalk (Elastic Beanstalk) — Elastic [Beanstalk menyertakan izin X-Ray di profil instans defaultnya](#). Anda dapat menggunakan profil instans default, atau menambahkan izin tulis ke profil instans kustom.
- AWS Lambda (Lambda) — Tambahkan izin tulis ke peran eksekusi fungsi Anda.

Cara membuat peran untuk digunakan dengan X-Ray

1. Buka [konsol IAM](#).
2. Pilih Peran.
3. Pilih Buat Peran Baru.
4. Untuk Nama peran, ketik **xray-application**. Pilih Langkah Selanjutnya.
5. Untuk Jenis Peran, pilih Amazon EC2.
6. Lampirkan kebijakan terkelola berikut untuk memberikan akses aplikasi Anda ke Layanan AWS:

- **AWSXRayDaemonWriteAccess**— Memberikan izin daemon X-Ray untuk mengunggah data jejak.

Jika aplikasi Anda menggunakan AWS SDK untuk mengakses layanan lain, tambahkan kebijakan yang memberikan akses ke layanan tersebut.

7. Pilih Langkah Selanjutnya.

8. Pilih Buat Peran.

Izin pengguna untuk enkripsi

X-Ray mengenkripsi semua data pelacakan dan secara default, dan Anda dapat [mengonfigurasinya untuk menggunakan kunci yang Anda kelola](#). Jika Anda memilih kunci yang dikelola AWS Key Management Service pelanggan, Anda perlu memastikan bahwa kebijakan akses kunci memungkinkan Anda memberikan izin kepada X-Ray untuk menggunakannya untuk mengenkripsi. Pengguna lain di akun Anda juga memerlukan akses ke kunci untuk melihat data pelacakan terenkripsi di konsol X-Ray.

Untuk kunci yang dikelola pelanggan, konfigurasi kunci Anda dengan kebijakan akses yang memungkinkan tindakan berikut:

- Pengguna yang mengonfigurasi kunci dalam X-Ray memiliki izin untuk memanggil `kms:CreateGrant` dan `kms:DescribeKey`.
- Pengguna yang dapat mengakses data pelacakan terenkripsi memiliki izin untuk memanggil `kms:Decrypt`.

Saat Anda menambahkan pengguna ke grup Pengguna kunci di bagian konfigurasi kunci dari konsol IAM, mereka memiliki izin untuk kedua operasi ini. Izin hanya perlu disetel pada kebijakan utama, sehingga Anda tidak memerlukan AWS KMS izin apa pun pada pengguna, grup, atau peran Anda. Untuk informasi selengkapnya, lihat [Menggunakan Kebijakan Utama di Panduan AWS KMS Pengembang](#).

Untuk enkripsi default, atau jika Anda memilih CMK AWS terkelola (`aws/xray`), izin didasarkan pada siapa yang memiliki akses ke APIs X-Ray. Siapa saja yang memiliki akses ke [PutEncryptionConfig](#), termasuk dalam `AWSXrayFullAccess`, dapat mengubah konfigurasi enkripsi. Untuk mencegah pengguna mengubah kunci enkripsi, jangan beri mereka izin untuk menggunakan [PutEncryptionConfig](#).

AWS X-Ray contoh kebijakan berbasis identitas

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya X-Ray. Mereka juga tidak dapat melakukan tugas menggunakan AWS Management Console, AWS CLI, atau AWS API. Administrator harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya tertentu yang mereka butuhkan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat [Membuat Kebijakan pada Tab JSON](#) dalam Panduan Pengguna IAM.

Topik

- [Praktik terbaik kebijakan](#)
- [Menggunakan konsol X-Ray](#)
- [Izinkan para pengguna untuk melihat izin mereka sendiri](#)
- [Mengelola akses ke grup X-Ray dan aturan pengambilan sampel berdasarkan tanda](#)
- [Kebijakan terkelola IAM untuk X-Ray](#)
- [Pembaruan X-Ray ke kebijakan AWS terkelola](#)
- [Menentukan sumber daya dalam kebijakan IAM](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya X-Ray di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Kebijakan yang dikelola AWS](#) atau [Kebijakan yang dikelola AWS untuk fungsi tugas](#) dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya

dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat [Kebijakan dan izin dalam IAM](#) dalam Panduan Pengguna IAM.

- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Kondisi](#) dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat [Validasi kebijakan dengan IAM Access Analyzer](#) dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat [Amankan akses API dengan MFA](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat [Praktik terbaik keamanan di IAM](#) dalam Panduan Pengguna IAM.

Menggunakan konsol X-Ray

Untuk mengakses AWS X-Ray konsol, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya X-Ray di Akun AWS. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Untuk memastikan bahwa entitas tersebut masih dapat menggunakan konsol X-Ray, lampirkan kebijakan `AWSXRayReadOnlyAccess` AWS terkelola ke entitas. Kebijakan ini dijelaskan secara

lebih rinci dalam [kebijakan yang dikelola IAM untuk X-Ray](#). Untuk informasi selengkapnya, lihat [Menambahkan Izin ke Pengguna](#) dalam Panduan Pengguna IAM.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai alternatif, hanya izinkan akses ke tindakan yang cocok dengan operasi API yang sedang Anda coba lakukan.

Izinkan para pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. 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": "*"
    }
  ]
}
```

```

    }
  ]
}

```

Mengelola akses ke grup X-Ray dan aturan pengambilan sampel berdasarkan tanda

Anda dapat menggunakan syarat dalam kebijakan berbasis identitas Anda untuk mengontrol akses ke grup X-Ray dan aturan pengambilan sampel berdasarkan tanda. Contoh kebijakan berikut dapat digunakan untuk menolak peran pengguna izin untuk membuat, menghapus, atau memperbarui grup dengan tag `stage:prod` atau `stage:preprod`. Untuk informasi selengkapnya tentang penandaan aturan dan grup pengambilan sampel X-Ray, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#).

Untuk menolak akses pengguna untuk membuat, memperbarui, atau menghapus grup dengan tanda `stage:prod` atau `stage:preprod`, tetapkan peran pengguna dengan kebijakan yang mirip dengan berikut ini.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllXRay",
      "Effect": "Allow",
      "Action": "xray:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyCreateGroupWithStage",
      "Effect": "Deny",
      "Action": [
        "xray:CreateGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/stage": [
            "preprod",
            "prod"
          ]
        }
      }
    }
  ],
}

```

```

{
  "Sid": "DenyUpdateGroupWithStage",
  "Effect": "Deny",
  "Action": [
    "xray:UpdateGroup",
    "xray>DeleteGroup"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/stage": [
        "preprod",
        "prod"
      ]
    }
  }
}
]
}

```

Untuk menolak pembuatan aturan pengambilan sampel, gunakan `aws:RequestTag` untuk menunjukkan tanda yang tidak dapat dilewatkan sebagai bagian dari permintaan pembuatan. Untuk menolak pembaruan atau penghapusan aturan pengambilan sampel, gunakan `aws:ResourceTag` untuk menolak tindakan berdasarkan tanda pada sumber daya tersebut.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllXRay",
      "Effect": "Allow",
      "Action": "xray:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyCreateSamplingRuleWithStage",
      "Effect": "Deny",
      "Action": "xray:CreateSamplingRule",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/stage": [
            "preprod",

```

```

        "prod"
      ]
    }
  },
  {
    "Sid": "DenyUpdateSamplingRuleWithStage",
    "Effect": "Deny",
    "Action": [
      "xray:UpdateSamplingRule",
      "xray:DeleteSamplingRule"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/stage": [
          "preprod",
          "prod"
        ]
      }
    }
  }
]
}

```

Anda dapat melampirkan kebijakan ini (atau menggabungkannya ke dalam satu kebijakan, lalu melampirkan kebijakan) ke pengguna di akun Anda. Bagi pengguna untuk membuat perubahan ke grup atau aturan pengambilan sampel, grup atau aturan pengambilan sampel tidak boleh ditandai `stage=prepod` atau `stage=prod`. Kunci tanda syarat Stage cocok dengan kedua Stage dan stage karena nama kunci syarat tidak terpengaruh huruf besar/kecil. Untuk informasi selengkapnya tentang blok syarat, lihat [Elemen Kebijakan JSON IAM: Syarat](#) dalam Panduan Pengguna IAM.

Pengguna dengan peran yang memiliki kebijakan berikut terlampir tidak dapat menambahkan tanda `role:admin` ke sumber daya, dan tidak dapat menghapus tanda dari sumber daya yang `role:admin` terkait dengan hal tersebut.

```

{
  "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"
            }
        }
    }
]
}

```

Kebijakan terkelola IAM untuk X-Ray

Untuk membuat pemberian izin mudah, IAM mendukung kebijakan terkelola untuk setiap layanan. Layanan dapat memperbarui kebijakan terkelola ini dengan izin baru saat merilis yang baru APIs. AWS X-Ray menyediakan kebijakan terkelola untuk kasus penggunaan hanya baca, tulis, dan administrator.

- **AWSXrayReadOnlyAccess**— Baca izin untuk menggunakan konsol X-Ray, AWS CLI, atau AWS SDK untuk mendapatkan data jejak, peta jejak, wawasan, dan konfigurasi X-Ray dari X-Ray API. [Termasuk Observability Access Manager \(OAM\) `oam:ListSinks` dan `oam:ListAttachedSinks` izin untuk memungkinkan konsol melihat jejak yang dibagikan dari akun sumber sebagai bagian dari CloudWatch pengamatan lintas akun.](#) Tindakan `BatchGetTraceSummaryById` dan `GetDistinctTraceGraphs` API tidak dimaksudkan untuk dipanggil oleh kode Anda, dan tidak termasuk dalam AWS CLI dan 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**— Tulis izin untuk menggunakan daemon X-Ray, AWS CLI, atau AWS SDK untuk mengunggah dokumen segmen dan telemetri ke X-Ray API. Memuat izin baca untuk mendapatkan [aturan pengambilan sampel](#) dan melaporkan hasil pengambilan sampel.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- **AWSXrayCrossAccountSharingConfiguration**— Memberikan izin untuk membuat, mengelola, dan melihat tautan Pengelola Akses Pengamatan untuk berbagi sumber daya X-Ray antar akun. Digunakan untuk mengaktifkan [observabilitas CloudWatch lintas akun](#) antara akun sumber dan pemantauan.

```
{
  "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**— Izin untuk menggunakan semua X-Ray APIs, termasuk izin baca, izin tulis, dan izin untuk mengonfigurasi pengaturan kunci enkripsi dan aturan pengambilan sampel. [Termasuk Observability Access Manager \(OAM\) `oam:ListSinks` dan `oam:ListAttachedSinks` izin untuk memungkinkan konsol melihat jejak yang dibagikan dari akun sumber sebagai bagian dari CloudWatch pengamatan lintas akun.](#)

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:*",
                "oam:ListSinks"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [

```

```

        "oam:ListAttachedLinks"
      ],
      "Resource": "arn:aws:oam:*:*:sink/*"
    }
  ]
}

```

Untuk menambahkan kebijakan terkelola ke pengguna IAM, grup IAM, atau IAM role.

1. Buka [konsol IAM](#).
2. Buka peran yang terkait dengan profil instans, pengguna IAM, atau grup IAM.
3. Di Izin, lampirkan kebijakan terkelola.

Pembaruan X-Ray ke kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk X-Ray sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman [riwayat Dokumen](#) X-Ray.

Perubahan	Deskripsi	Tanggal
IAM mengelola kebijakan untuk X-Ray - Ditambahkan baruAWSXrayCrossAccountSharingConfiguration, dan diperbarui AWSXrayReadOnlyAccess dan AWSXrayFullAccess kebijakan.	X-Ray menambahkan izin Observability Access Manager (OAM) oam:ListSinks dan kebijakan ini oam:ListAttachedSinks untuk memungkinkan konsol melihat jejak yang dibagikan dari akun sumber sebagai bagian dari CloudWatch pengamatan lintas akun.	27 November 2022
Kebijakan terkelola IAM untuk X-Ray - Pembaruan ke AWSXrayReadOnlyAccess kebijakan.	X-Ray menambahkan aksi API,ListResourcePolicies.	15 November 2022

Perubahan	Deskripsi	Tanggal
Menggunakan konsol X-Ray — Perbarui ke AWSXrayResourceOnlyAccess kebijakan	X-Ray menambahkan dua tindakan API baru, BatchGetTraceSummaryById dan GetDistinctTraceGraphs . Tindakan ini tidak dimaksudkan untuk dipanggil oleh kode Anda. Oleh karena itu, tindakan API ini tidak termasuk dalam AWS CLI dan AWS SDKs.	11 November 2022

Menentukan sumber daya dalam kebijakan IAM

Anda dapat mengontrol akses ke sumber daya menggunakan kebijakan IAM. Untuk tindakan yang mendukung sumber daya tingkat izin, Anda menggunakan Amazon Resource Name (ARN) untuk mengidentifikasi sumber daya yang diberlakukan oleh kebijakan tersebut.

Semua tindakan X-Ray dapat digunakan dalam kebijakan IAM untuk memberikan atau menolak izin pengguna untuk menggunakan tindakan tersebut. Namun, tidak semua [tindakan X-Ray](#) mendukung izin tingkat sumber daya, yang memungkinkan Anda untuk menentukan sumber daya tempat tindakan dapat dilakukan.

Untuk tindakan yang tidak mendukung izin tingkat sumber daya, Anda harus menggunakan "*" sebagai sumber daya.

Tindakan X-Ray berikut mendukung izin tingkat sumber daya:

- CreateGroup
- GetGroup
- UpdateGroup
- DeleteGroup
- CreateSamplingRule
- UpdateSamplingRule

- DeleteSamplingRule

Berikut ini adalah contoh kebijakan izin berbasis identitas untuk tindakan `CreateGroup`. Contoh ini menunjukkan penggunaan ARN yang berkaitan dengan nama Grup `local-users` dengan ID unik sebagai wildcard. ID unik terbuat ketika grup dibuat, sehingga tidak dapat diprediksi di kebijakan sebelumnya. Saat menggunakan `GetGroup`, `UpdateGroup`, atau `DeleteGroup`, Anda dapat menentukan ini sebagai wildcard atau ARN yang tepat, termasuk ID.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:CreateGroup"
      ],
      "Resource": [
        "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
      ]
    }
  ]
}
```

Berikut ini adalah contoh kebijakan izin berbasis identitas untuk tindakan `CreateSamplingRule`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:CreateSamplingRule"
      ],
      "Resource": [
        "arn:aws:xray:eu-west-1:123456789012:sampling-rule/base-scorekeep"
      ]
    }
  ]
}
```

 Note

ARN dari aturan pengambilan sampel ditentukan berdasarkan namanya. Tidak seperti grup ARNs, aturan pengambilan sampel tidak memiliki ID yang dihasilkan secara unik.

Memecahkan masalah AWS X-Ray identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temukan saat bekerja dengan X-Ray dan IAM.

Topik

- [Saya tidak diotorisasi untuk melakukan tindakan di X-Ray](#)
- [Saya tidak berwenang untuk melakukan iam: PassRole](#)
- [Saya seorang administrator dan ingin mengizinkan orang lain mengakses X-Ray](#)
- [Saya ingin mengizinkan orang-orang di luar saya Akun AWS untuk mengakses sumber daya X-Ray saya](#)

Saya tidak diotorisasi untuk melakukan tindakan di X-Ray

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Contoh kesalahan berikut terjadi ketika mateojackson pengguna mencoba menggunakan konsol untuk melihat detail tentang aturan pengambilan sampel tetapi tidak memiliki `xray:GetSamplingRules` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
perform: xray:GetSamplingRules on resource: arn:${Partition}:xray:${Region}:
${Account}:sampling-rule/${SamplingRuleName}
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya agar dia dapat mengakses sumber daya aturan pengambilan sampel menggunakan tindakan `xray:GetSamplingRules`.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak berwenang untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui untuk memungkinkan Anda meneruskan peran ke X-Ray.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi saat pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di X-Ray. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya seorang administrator dan ingin mengizinkan orang lain mengakses X-Ray

Untuk memungkinkan orang lain mengakses X-Ray, Anda harus memberikan izin kepada orang atau aplikasi yang membutuhkan akses. Jika Anda menggunakan AWS IAM Identity Center untuk mengelola orang dan aplikasi, Anda menetapkan set izin kepada pengguna atau grup untuk menentukan tingkat akses mereka. Set izin secara otomatis membuat dan menetapkan kebijakan IAM ke peran IAM yang terkait dengan orang atau aplikasi. Untuk informasi selengkapnya, lihat [Set izin](#) di Panduan AWS IAM Identity Center Pengguna.

Jika Anda tidak menggunakan IAM Identity Center, Anda harus membuat entitas IAM (pengguna atau peran) untuk orang atau aplikasi yang membutuhkan akses. Anda kemudian harus melampirkan kebijakan ke entitas yang memberi izin yang tepat di X-Ray. Setelah izin diberikan, berikan kredensialnya kepada pengguna atau pengembang aplikasi. Mereka akan menggunakan kredensi tersebut untuk mengakses AWS. Untuk mempelajari selengkapnya tentang membuat pengguna, grup, kebijakan, dan izin IAM, lihat [Identitas dan Kebijakan IAM dan izin di IAM di Panduan Pengguna IAM](#).

Saya ingin mengizinkan orang-orang di luar saya Akun AWS untuk mengakses sumber daya X-Ray saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah X-Ray mendukung fitur-fitur ini, lihat [Bagaimana AWS X-Ray bekerja dengan IAM](#).
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

Penebangan dan pemantauan di AWS X-Ray

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan performa solusi AWS Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. AWS menyediakan beberapa alat untuk memantau sumber daya X-Ray Anda dan menanggapi potensi insiden:

AWS CloudTrail Log

AWS X-Ray terintegrasi dengan AWS CloudTrail untuk merekam tindakan API yang dibuat oleh pengguna, peran, atau AWS layanan di X-Ray. Anda dapat menggunakan CloudTrail untuk memantau permintaan X-Ray API secara real time dan menyimpan log di Amazon S3, Amazon

CloudWatch Log, dan Amazon Events. CloudWatch Untuk informasi selengkapnya, lihat [Mencatat panggilan X-Ray API dengan AWS CloudTrail](#).

AWS Config Melacak

AWS X-Ray terintegrasi dengan AWS Config untuk merekam perubahan konfigurasi yang dibuat pada sumber daya enkripsi X-Ray Anda. Anda dapat menggunakan AWS Config inventaris sumber daya enkripsi X-Ray, mengaudit riwayat konfigurasi X-Ray, dan mengirim pemberitahuan berdasarkan perubahan sumber daya. Untuk informasi selengkapnya, lihat [Melacak perubahan konfigurasi enkripsi X-Ray AWS Config](#).

CloudWatch Pemantauan Amazon

Anda dapat menggunakan X-Ray SDK for Java untuk mempublikasikan metrik CloudWatch Amazon tanpa sampel dari segmen X-Ray yang dikumpulkan. Metrik ini berasal dari waktu mulai dan berakhir segmen, serta tanda status error, kesalahan, dan bendera status throttled. Gunakan metrik pelacakan ini untuk mengekspos percobaan ulang dan masalah ketergantungan dalam subsegmen. Untuk informasi selengkapnya, lihat [AWS X-Ray metrik untuk X-Ray SDK for Java](#).

Validasi kepatuhan untuk AWS X-Ray

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat [Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS](#) dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat [Program AWS Kepatuhan Program AWS](#) .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifact](#) .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- [Kepatuhan dan Tata Kelola Keamanan](#) – Panduan implementasi solusi ini membahas pertimbangan arsitektur serta memberikan langkah-langkah untuk menerapkan fitur keamanan dan kepatuhan.
- [Referensi Layanan yang Memenuhi Syarat HIPAA](#) — Daftar layanan yang memenuhi syarat HIPAA. Tidak semua memenuhi Layanan AWS syarat HIPAA.
- [AWS Sumber Daya AWS](#) — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.

- [AWS Panduan Kepatuhan Pelanggan](#) - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).
- [Mengevaluasi Sumber Daya dengan Aturan](#) dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- [AWS Security Hub](#)— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat [Referensi kontrol Security Hub](#).
- [Amazon GuardDuty](#) — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- [AWS Audit Manager](#)Ini Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Ketahanan di AWS X-Ray

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat [Infrastruktur AWS Global](#).

Keamanan infrastruktur di AWS X-Ray

Sebagai layanan terkelola, AWS X-Ray dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat [Keamanan AWS Cloud](#). Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat [Perlindungan Infrastruktur dalam Kerangka Kerja](#) yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses X-Ray melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda bisa menggunakan [AWS Security Token Service](#) (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Menggunakan AWS X-Ray dengan titik akhir VPC

Jika Anda menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk meng-host AWS sumber daya Anda, Anda dapat membuat koneksi pribadi antara VPC dan X-Ray. Ini memungkinkan sumber daya di Amazon VPC Anda untuk berkomunikasi dengan layanan X-Ray tanpa melalui internet publik.

Amazon VPC adalah Layanan AWS yang dapat Anda gunakan untuk meluncurkan AWS sumber daya di jaringan virtual yang Anda tentukan. Dengan VPC, Anda memiliki kendali terhadap pengaturan jaringan, seperti rentang alamat IP, subnet, tabel rute, dan gateway jaringan. Untuk menghubungkan VPC Anda ke X-Ray, Anda menentukan [antarmuka VPC endpoint](#). Titik akhir menyediakan konektivitas yang andal dan dapat diskalakan ke X-Ray tanpa memerlukan gateway internet, instans terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi selengkapnya, lihat [Apa yang dimaksud dengan Amazon VPC](#) dalam Panduan Pengguna Amazon VPC.

Endpoint VPC antarmuka didukung oleh AWS PrivateLink, sebuah AWS teknologi yang memungkinkan komunikasi pribadi antara Layanan AWS dengan menggunakan antarmuka network

elastis dengan alamat IP pribadi. Untuk informasi selengkapnya, lihat [Baru — AWS PrivateLink untuk posting Layanan AWS blog](#) dan [Memulai](#) di Panduan Pengguna Amazon VPC.

Untuk memastikan Anda dapat membuat titik akhir VPC untuk X-Ray pada pilihan Wilayah AWS Anda, lihat. [Wilayah yang Didukung](#)

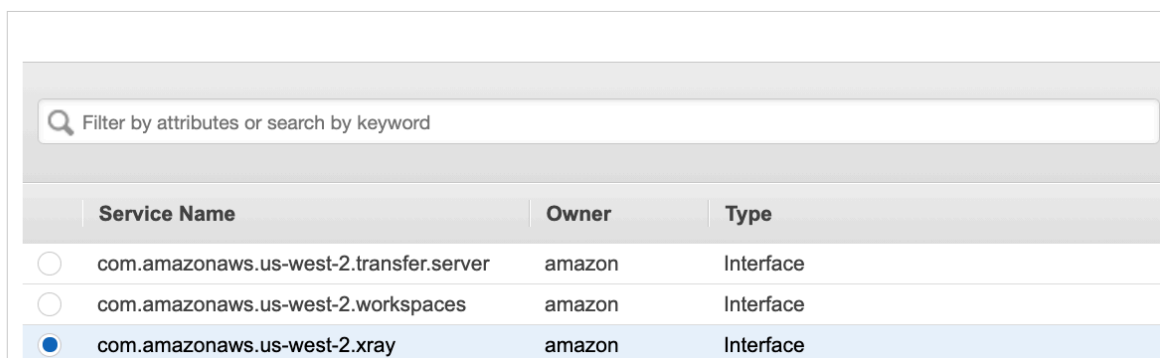
Membuat VPC endpoint untuk X-Ray

Untuk mulai menggunakan X-Ray dengan VPC Anda, buat VPC endpoint antarmuka untuk X-Ray.

1. Buka konsol Amazon VPC di. <https://console.aws.amazon.com/vpc/>
2. Navigasi ke Titik akhir dalam panel navigasi dan pilih Buat Titik akhir.
3. Cari dan pilih nama AWS X-Ray layanan: `com.amazonaws.region.xray`.

Service category ☒ AWS services
☐ Find service by name
☐ Your AWS Marketplace services

Service Name `com.amazonaws.us-west-2.xray` ⓘ



	Service Name	Owner	Type
☐	com.amazonaws.us-west-2.transfer.server	amazon	Interface
☐	com.amazonaws.us-west-2.workspaces	amazon	Interface
☒	com.amazonaws.us-west-2.xray	amazon	Interface

4. Pilih VPC yang Anda inginkan lalu pilih subnet di VPC Anda untuk menggunakan titik akhir antarmuka. Antarmuka jaringan titik akhir dibuat di subnet yang dipilih. Anda dapat menentukan lebih dari satu subnet di Availability Zone yang berbeda (sebagaimana didukung oleh layanan) untuk membantu memastikan bahwa titik akhir antarmuka Anda tahan terhadap kegagalan Availability Zone. Jika Anda melakukannya, antarmuka jaringan antarmuka dibuat di setiap subnet yang Anda tentukan..

VPC*  

Subnets 

Availability Zone	Subnet ID
☒ us-west-2a (usw2-az1)	subnet-40d87938
☐ us-west-2b (usw2-az2)	subnet-ff4281b5
☐ us-west-2c (usw2-az3)	subnet-d14bfb8c
☐ us-west-2d (usw2-az4)	subnet-1faf8734

- (Optional) DNS Privat diaktifkan secara default untuk titik akhir, sehingga Anda dapat membuat permintaan ke X-Ray menggunakan nama host DNS default-nya. Anda dapat memilih untuk menonaktifkannya.
- Tentukan grup keamanan yang akan dikaitkan dengan antarmuka jaringan titik akhir.

Security group [Create a new security group](#) 

Select security groups ▲

1 to 5 of 5

☐	Group ID	Group Name	VPC ID		Description	Owner ID
☐	sg-0683c...	ssh-http	vpc-4f6e3a37	EC2-VPC	launch-wizar...	979300271395
☐	sg-0774...	awseb-e-7xv5...	vpc-4f6e3a37	EC2-VPC	SecurityGrou...	979300271395
☐	sg-0a46...	launch-wizard-1	vpc-4f6e3a37	EC2-VPC	launch-wizar...	979300271395
☐	sg-0d62...	awseb-e-7xv5...	vpc-4f6e3a37	EC2-VPC	Elastic Beans...	979300271395
☒	sg-d4f14...	default	vpc-4f6e3a37	EC2-VPC	default VPC s...	979300271395

Close

- (Optional) Tentukan kebijakan khusus untuk mengendalikan izin untuk mengakses layanan X-Ray. Secara default, akses penuh diizinkan.

Mengendalikan akses ke X-Ray VPC Endpoint Anda

Kebijakan VPC endpoint adalah kebijakan sumber daya IAM yang Anda lampirkan ke titik akhir ketika membuat atau mengubah titik akhir. Jika Anda tidak melampirkan kebijakan ketika membuat titik akhir, Amazon VPC melampirkan kebijakan default untuk Anda sehingga memungkinkan akses penuh ke layanan. Kebijakan titik akhir tidak membatalkan atau mengganti kebijakan pengguna IAM

atau kebijakan khusus layanan. Ini adalah kebijakan terpisah untuk mengendalikan akses dari titik akhir ke layanan tertentu. Kebijakan titik akhir harus ditulis dalam format JSON. Untuk informasi selengkapnya, lihat [Mengendalikan Akses ke Layanan dengan VPC Endpoint](#) dalam Panduan Pengguna Amazon VPC.

Kebijakan VPC endpoint memungkinkan Anda mengontrol izin untuk berbagai tindakan X-Ray. Misalnya, Anda dapat membuat kebijakan untuk mengizinkan saja PutTraceSegment dan menolak semua tindakan lainnya. Ini membatasi beban kerja dan layanan di VPC untuk hanya mengirim data pelacakan ke X-Ray dan menolak tindakan lain seperti mengambil data, mengubah konfigurasi enkripsi, atau membuat/memperbarui grup.

Berikut ini adalah contoh kebijakan titik akhir untuk X-Ray. Kebijakan ini mengizinkan pengguna yang terhubung ke X-Ray melalui VPC untuk mengirim data segmen ke X-Ray, dan juga mencegah mereka melakukan tindakan X-Ray lainnya.

```
{
  "Statement": [
    {
      "Sid": "Allow PutTraceSegments",
      "Principal": "*",
      "Action": [
        "xray:PutTraceSegments"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Untuk mengedit kebijakan VPC endpoint untuk X-Ray

1. Buka konsol Amazon VPC di <https://console.aws.amazon.com/vpc/>
2. Di panel navigasi, pilih Titik akhir.
3. Jika Anda belum membuat titik akhir untuk X-Ray, ikuti langkah-langkah di [Membuat VPC endpoint untuk X-Ray](#).
4. Pilih com.amazonaws.*region*.xray endpoint, lalu pilih tab Policy.
5. Pilih Edit Kebijakan, lalu lakukan perubahan.

Wilayah yang Didukung

X-Ray saat ini mendukung titik akhir VPC sebagai berikut: Wilayah AWS

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Afrika (Cape Town)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)
- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Milan)
- Eropa (Paris)
- Eropa (Stockholm)
- Timur Tengah (Bahrain)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (AS-Timur)
- AWS GovCloud (AS-Barat)

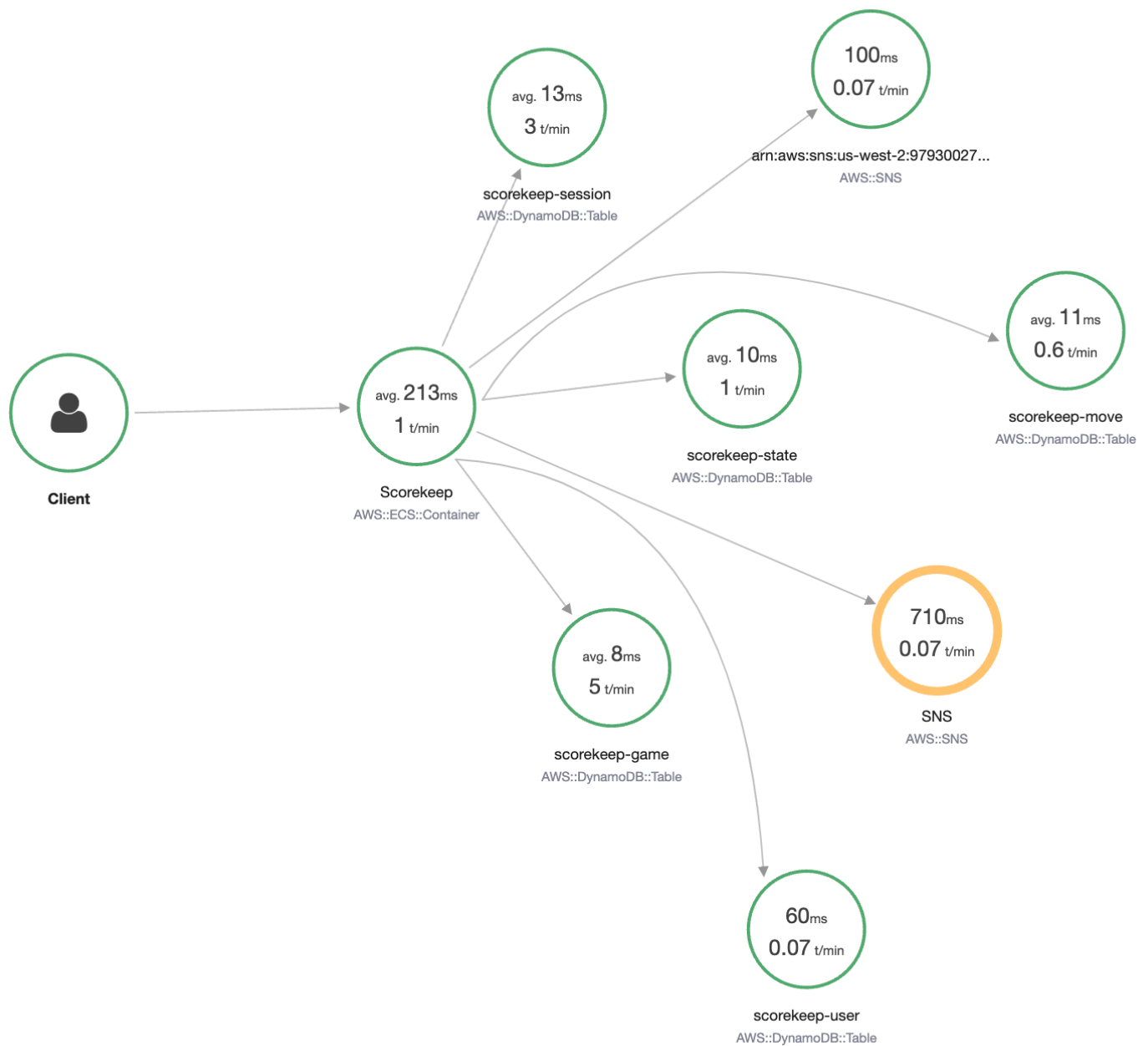
AWS X-Ray aplikasi sampel

Aplikasi [eb-java-scorekeep](#) sampel AWS X-Ray, tersedia di GitHub, menunjukkan penggunaan AWS X-Ray SDK untuk instrumen panggilan HTTP yang masuk, klien SDK DynamoDB, dan klien HTTP. Aplikasi sampel digunakan AWS CloudFormation untuk membuat tabel DynamoDB, mengkompilasi kode Java pada instance, dan menjalankan daemon X-Ray tanpa konfigurasi tambahan.

Lihat [tutorial Scorekeep](#) untuk mulai menginstal dan menggunakan aplikasi sampel berinstrumen, menggunakan atau file. AWS Management Console AWS CLI



Sampel mencakup aplikasi web front-end, API yang dipanggilnya, dan tabel DynamoDB yang digunakan untuk menyimpan data. Instrumentasi dasar dengan [filter](#), [plugin](#), dan [klien AWS SDK berinstrumen](#) ditampilkan di cabang proyek. `xray-gettingstarted` ini adalah cabang yang Anda deploy di [tutorial memulai](#). Karena cabang ini hanya mencakup instrumentasi dasar, Anda dapat membedakannya dengan cabang master agar dapat memahami instrumentasi dasar dengan cepat.



Aplikasi sampel menunjukkan instrumentasi dasar dalam file ini:

- Filter permintaan HTTP – [WebConfig.java](#)
- AWS Instrumentasi klien SDK — [build.gradle](#)

xrayCabang aplikasi mencakup penggunaan HTTPClient, Anotasi, kueri SQL, subsegmen kustom, AWS Lambdafungsi instrumentasi, dan kode inisialisasi instrumentasi dan skrip.

Untuk mendukung login dan AWS SDK untuk JavaScript penggunaan pengguna di browser, `xray-cognito` cabang menambahkan Amazon Cognito untuk mendukung otentikasi dan otorisasi pengguna. Dengan kredensial yang diambil dari Amazon Cognito, aplikasi web juga mengirimkan data pelacakan ke X-Ray untuk mencatat informasi permintaan dari sudut pandang klien. Klien browser muncul sebagai simpulnya sendiri di peta jejak, dan mencatat informasi tambahan, termasuk URL halaman yang dilihat pengguna, dan ID pengguna.

Akhirnya, cabang `xray-worker` menambahkan fungsi Python Lambda yang diinstrumentasi yang berjalan secara independen, memproses item dari antrean Amazon SQS. Scorekeep menambahkan item ke antrean setiap kali game berakhir. Pekerja Lambda, yang dipicu oleh CloudWatch Peristiwa, menarik item dari antrian setiap beberapa menit dan memprosesnya untuk menyimpan catatan game di Amazon S3 untuk dianalisis.

Topik

- [Memulai dengan aplikasi sampel Scorekeep](#)
- [Menginstrumentasi klien AWS SDK secara manual](#)
- [Membuat subsegmen tambahan](#)
- [Merekam anotasi, metadata, dan pengguna IDs](#)
- [Instrumentasi panggilan HTTP keluar](#)
- [Menginstrumentasi panggilan ke basis data PostgreSQL](#)
- [Fungsi instrumentasi AWS Lambda](#)
- [Menginstrumentasi kode perusahaan rintisan](#)
- [Skrip instrumentasi](#)
- [Menginstrumentasi klien aplikasi web](#)
- [Menggunakan klien berinstrumen di utas pekerja](#)

Memulai dengan aplikasi sampel Scorekeep

Tutorial ini menggunakan `xray-gettingstarted` cabang [aplikasi sampel Scorekeep](#), yang digunakan AWS CloudFormation untuk membuat dan mengonfigurasi sumber daya yang menjalankan aplikasi sampel dan daemon X-Ray di Amazon ECS. Aplikasi ini menggunakan framework Spring untuk mengimplementasikan API web JSON dan AWS SDK untuk Java untuk mempertahankan data ke Amazon DynamoDB. Filter servlet dalam instrumen aplikasi semua permintaan masuk yang dilayani oleh aplikasi, dan penanganan permintaan pada instrumen klien AWS SDK panggilan hilir ke DynamoDB.

Anda dapat mengikuti tutorial ini menggunakan salah satu AWS Management Console atau AWS CLI.

Bagian-bagian

- [Prasyarat](#)
- [Instal aplikasi Scorekeep menggunakan CloudFormation](#)
- [Hasilkan data pelacakan](#)
- [Lihat peta jejak di AWS Management Console](#)
- [Mengonfigurasi notifikasi Amazon SNS](#)
- [Jelajahi aplikasi sampel](#)
- [Opsional: Kebijakan hak istimewa terbatas](#)
- [Bersihkan](#)
- [Langkah selanjutnya](#)

Prasyarat

Tutorial ini digunakan AWS CloudFormation untuk membuat dan mengkonfigurasi sumber daya yang menjalankan aplikasi sampel dan daemon X-Ray. Prasyarat berikut diperlukan untuk menginstal dan menjalankan tutorial:

1. Jika Anda menggunakan pengguna IAM dengan izin terbatas, tambahkan kebijakan pengguna berikut di konsol [IAM](#):
 - `AWSCloudFormationFullAccess`— untuk mengakses dan menggunakan CloudFormation
 - `AmazonS3FullAccess`— untuk mengunggah file template untuk CloudFormation menggunakan AWS Management Console
 - `IAMFullAccess`— untuk membuat peran EC2 instans Amazon ECS dan Amazon
 - `AmazonEC2FullAccess`— untuk membuat EC2 sumber daya Amazon
 - `AmazonDynamoDBFullAccess`— untuk membuat tabel DynamoDB
 - `AmazonECS_FullAccess`— untuk membuat sumber daya Amazon ECS
 - `AmazonSNSFullAccess`— untuk membuat topik Amazon SNS
 - `AWSXrayReadOnlyAccess`— untuk izin untuk melihat peta jejak dan jejak di konsol X-Ray
2. Untuk menjalankan tutorial menggunakan AWS CLI, [instal CLI](#) versi 2.7.9 atau yang lebih baru, dan konfigurasikan [CLI](#) dengan pengguna dari langkah sebelumnya. Pastikan wilayah

dikonfigurasi saat mengonfigurasi AWS CLI dengan pengguna. Jika suatu wilayah tidak dikonfigurasi, Anda harus menambahkan `--region` **AWS-REGION** ke setiap perintah CLI.

3. Pastikan bahwa [Git](#) diinstal, untuk mengkloning repo aplikasi sampel.
4. Gunakan contoh kode berikut untuk mengkloning `xray-gettingstarted` cabang repositori Scorekeep:

```
git clone https://github.com/aws-samples/eb-java-scorekeep.git xray-scorekeep -b
xray-gettingstarted
```

Instal aplikasi Scorekeep menggunakan CloudFormation

AWS Management Console

Instal aplikasi sampel menggunakan AWS Management Console

1. Buka [konsol CloudFormation](#)
2. Pilih Buat tumpukan dan kemudian pilih Dengan sumber daya baru dari menu drop-down.
3. Di bagian Tentukan templat, pilih Unggah file templat.
4. Pilih file, navigasikan ke `xray-scorekeep/cloudformation` folder yang dibuat saat Anda mengkloning git repo, dan pilih file `cf-resources.yaml`
5. Pilih Next untuk melanjutkan.
6. Masukkan `scorekeep` ke dalam kotak teks nama Stack, lalu pilih Berikutnya di bagian bawah halaman untuk melanjutkan. Perhatikan bahwa sisa tutorial ini mengasumsikan tumpukan diberi nama `scorekeep`.
7. Gulir ke bagian bawah halaman Configure stack options dan pilih Next untuk melanjutkan.
8. Gulir ke bagian bawah halaman Tinjauan, pilih kotak centang yang mengakui yang CloudFormation dapat membuat sumber daya IAM dengan nama khusus, dan pilih Buat tumpukan.
9. CloudFormation Tumpukan sekarang sedang dibuat. Status tumpukan akan `CREATE_IN_PROGRESS` sekitar lima menit sebelum diubah menjadi `CREATE_COMPLETE`. Status akan disegarkan secara berkala, atau Anda dapat me-refresh halaman.

AWS CLI

Instal aplikasi sampel menggunakan AWS CLI

1. Arahkan ke `cloudformation` folder `xray-scorekeep` repositori yang Anda kloning sebelumnya dalam tutorial:

```
cd xray-scorekeep/cloudformation/
```

2. Masukkan AWS CLI perintah berikut untuk membuat CloudFormation tumpukan:

```
aws cloudformation create-stack --stack-name scorekeep --capabilities  
"CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

3. Tunggu sampai status CloudFormation tumpukan `CREATE_COMPLETE`, yang akan memakan waktu sekitar lima menit. Gunakan AWS CLI perintah berikut untuk memeriksa status:

```
aws cloudformation describe-stacks --stack-name scorekeep --query  
"Stacks[0].StackStatus"
```

Hasilkan data pelacakan

Aplikasi sampel termasuk aplikasi web front-end. Gunakan aplikasi web untuk menghasilkan lalu lintas ke API dan mengirim data pelacakan ke X-Ray. Pertama, ambil URL aplikasi web menggunakan AWS Management Console atau: AWS CLI

AWS Management Console

Temukan URL aplikasi menggunakan AWS Management Console

1. Buka [konsol CloudFormation](#)
2. Pilih `scorekeep` tumpukan dari daftar.
3. Pilih tab `Output` pada halaman `scorekeep` tumpukan, dan pilih tautan `LoadBalancerUrl` URL untuk membuka aplikasi web.

AWS CLI

Temukan URL aplikasi menggunakan AWS CLI

1. Gunakan perintah berikut untuk menampilkan URL aplikasi web:

```
aws cloudformation describe-stacks --stack-name scorekeep --query  
"Stacks[0].Outputs[0].OutputValue"
```

2. Salin URL ini dan buka di browser untuk menampilkan aplikasi web Scorekeep.

Gunakan aplikasi web untuk menghasilkan data jejak

1. Pilih Buat untuk membuat pengguna dan sesi.
2. Ketik nama game, atur Aturan ke Tic Tac Toe, lalu pilih Buat untuk membuat game.
3. Pilih Mainkan untuk memulai game.
4. Pilih ubin untuk bergerak dan mengubah status game.

Masing-masing langkah ini menghasilkan permintaan HTTP ke API, dan panggilan hilir ke DynamoDB untuk membaca dan menulis pengguna, sesi, game, gerak, dan data status.

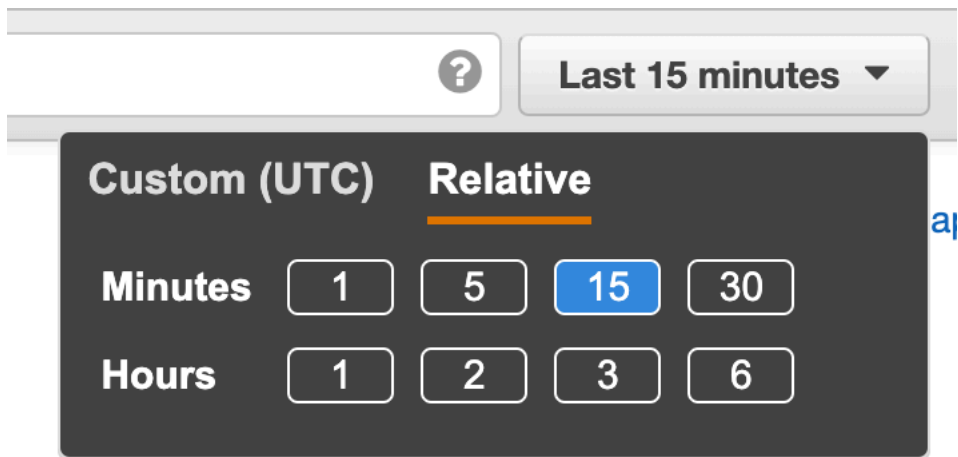
Lihat peta jejak di AWS Management Console

Anda dapat melihat peta jejak dan jejak yang dihasilkan oleh aplikasi sampel di X-Ray dan CloudWatch konsol.

X-Ray console

Gunakan konsol X-Ray

1. Buka halaman peta jejak [konsol X-Ray](#).
2. Konsol menunjukkan representasi grafik layanan yang dihasilkan X-Ray dari pelacakan data yang dikirim oleh aplikasi. Pastikan untuk menyesuaikan periode waktu peta jejak jika diperlukan, untuk memastikan bahwa itu akan menampilkan semua jejak sejak Anda pertama kali memulai aplikasi web.



?

Last 15 minutes ▼

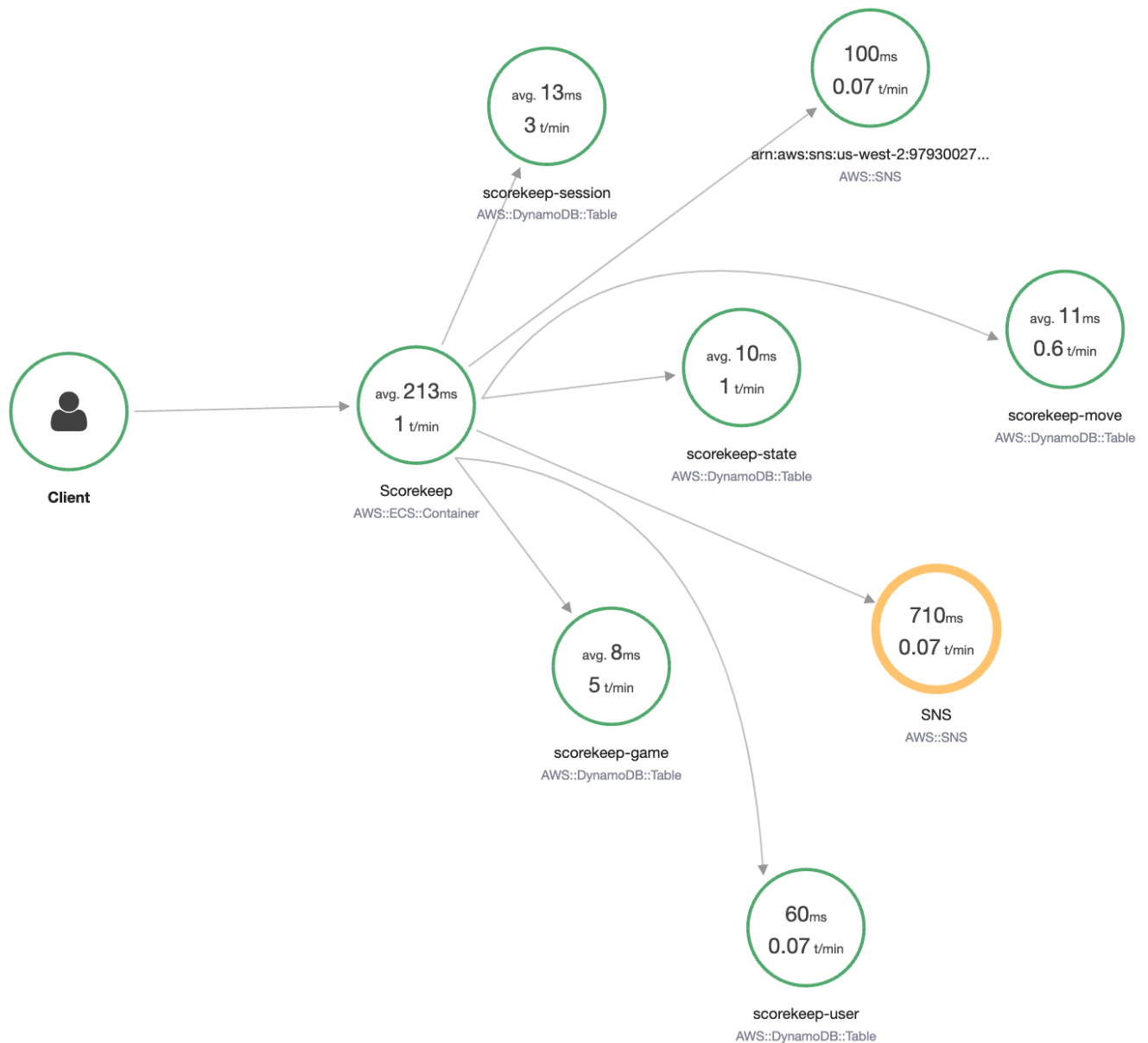
Custom (UTC) **Relative**

Minutes 1 5 15 30

Hours 1 2 3 6

Peta jejak menunjukkan klien aplikasi web, API yang berjalan di Amazon ECS, dan setiap tabel DynamoDB yang digunakan aplikasi. Setiap permintaan ke aplikasi, hingga jumlah maksimum permintaan yang dapat dikonfigurasi per detik, dilacak saat menyentuh API, menghasilkan permintaan ke layanan hilir, dan selesai.

Anda dapat memilih simpul apa pun dalam layanan grafik untuk melihat pelacakan untuk permintaan yang menghasilkan lalu lintas ke simpul tersebut. Saat ini, node Amazon SNS berwarna kuning. Telusuri paling detail untuk mencari tahu penyebabnya.



Untuk menemukan penyebab kesalahan

1. Pilih simpul bernama SNS. Panel detail simpul ditampilkan.
2. Pilih Lihat pelacakan untuk mengakses layar Gambaran umum pelacakan.
3. Pilih pelacakan dari Daftar pelacakan. Pelacakan ini tidak memiliki metode atau URL karena dicatat selama startup bukan sebagai respons permintaan masuk.

Q service("SNS") ? Last 5 Minutes ↺

Trace overview

Group by: URL

URL	Avg Latency	% of Traces	Response
-	1.3 sec	100.00%	1 OK, 0 Throttled, 0 Errors, 0 Faults

Trace list (1)

ID	Age	Method	Response	Latency	URL	Client IP	Annotations
...48b5a191	1.1 min			1.3 sec			0

- Pilih ikon status kesalahan dalam segmen Amazon SNS di bagian bawah halaman, untuk membuka halaman Pengecualian untuk subsegmen SNS.

[Traces](#) > Details

Q 1-62f40175-86b347fc50bc57a992e9b835 ? ↺

Timeline **Raw data**

Method	Response	Duration	Age	ID
--	--	2.1 sec	8.3 min (2022-08-10 19:05:25 UTC)	1-62f40175-86b347fc50bc57a992e9b835

▼ Trace Map

Client

AWS::EC2::Instance

Scorekeep

AWS::EC2::Instance

SNS

AWS::SNS

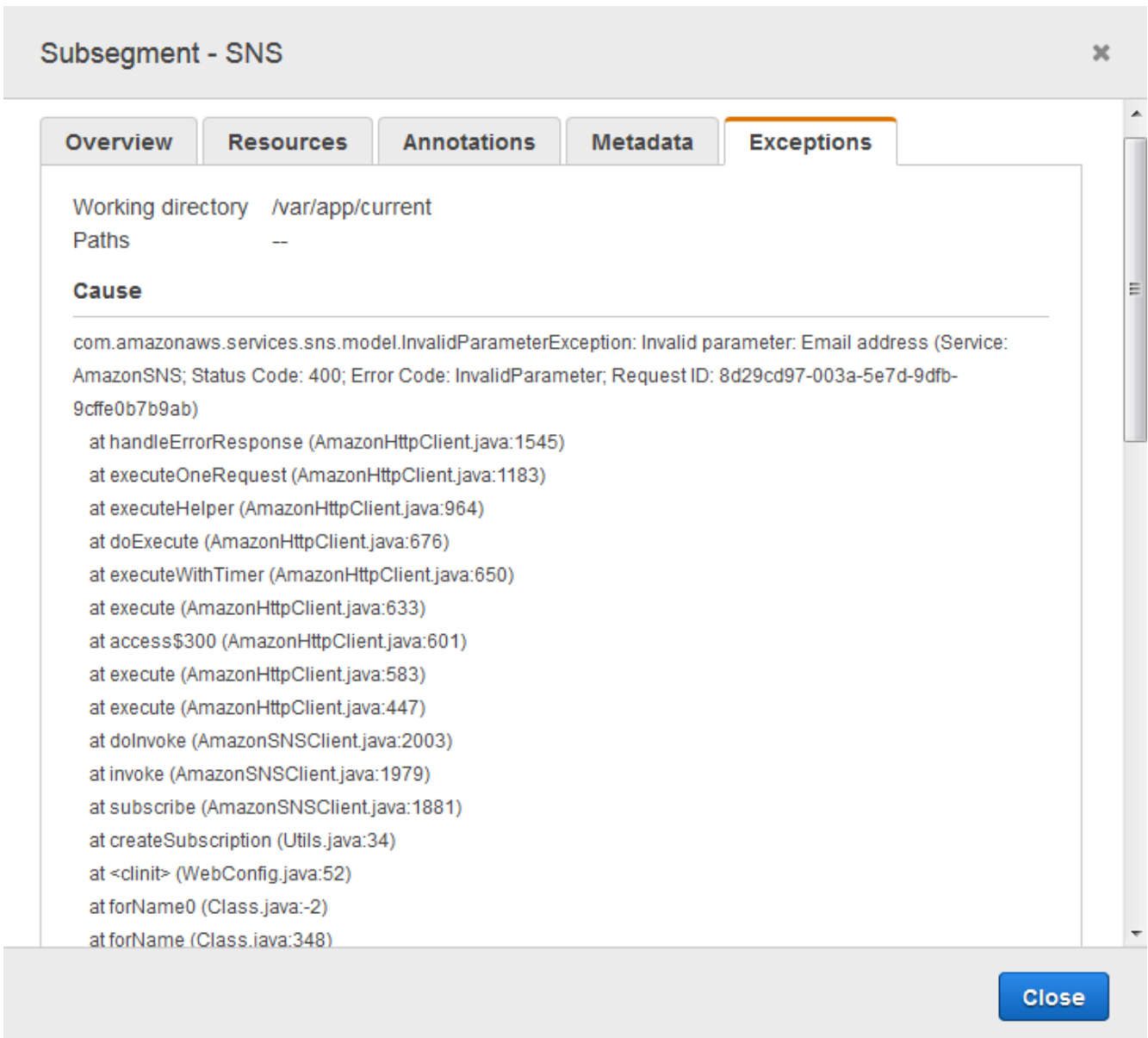
Services Icons ☒

None Health Traffic

No node resizing

Name	Res.	Duration	Status	0.0ms	200ms	400ms	600ms	800ms	1.0s	1.2s	1.4s	1.6s	1.8s	2.0s	2.2s
▼ Scorekeep AWS::EC2::Instance															
Scorekeep	-	2.1 sec	✓												
SNS	400	728 ms	⚠												
▶ SNS AWS::SNS (Client Response)															

5. X-Ray SDK secara otomatis menangkap pengecualian yang dilemparkan oleh instrumentasi AWS SDK klien dan mencatat pelacakan tumpukan.

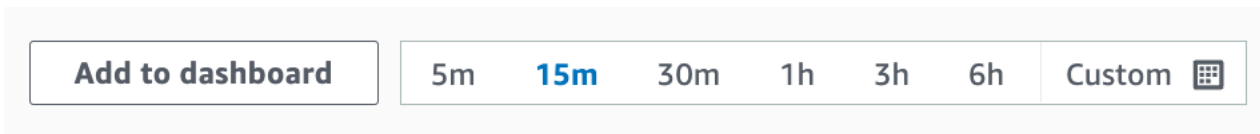


CloudWatch console

Gunakan CloudWatch konsol

1. Buka halaman [peta jejak X-Ray](#) CloudWatch konsol.
2. Konsol menunjukkan representasi grafik layanan yang dihasilkan X-Ray dari pelacakan data yang dikirim oleh aplikasi. Pastikan untuk menyesuaikan periode waktu peta jejak jika

diperlukan, untuk memastikan bahwa itu akan menampilkan semua jejak sejak Anda pertama kali memulai aplikasi web.



Peta jejak menunjukkan klien aplikasi web, API yang berjalan di Amazon EC2, dan setiap tabel DynamoDB yang digunakan aplikasi. Setiap permintaan ke aplikasi, hingga jumlah maksimum permintaan yang dapat dikonfigurasi per detik, dilacak saat menyentuh API, menghasilkan permintaan ke layanan hilir, dan selesai.

Anda dapat memilih simpul apa pun dalam layanan grafik untuk melihat pelacakan untuk permintaan yang menghasilkan lalu lintas ke simpul tersebut. Saat ini, node Amazon SNS berwarna oranye. Telusuri paling detail untuk mencari tahu penyebabnya.



Untuk menemukan penyebab kesalahan

1. Pilih simpul bernama SNS. Panel detail node SNS ditampilkan di bawah peta.
2. Pilih Lihat jejak untuk mengakses halaman Jejak.
3. Tambahkan bagian bawah halaman, pilih jejak dari daftar Jejak. Pelacakan ini tidak memiliki metode atau URL karena dicatat selama startup bukan sebagai respons permintaan masuk.

Traces [Info](#) 5m 15m **30m** 1h 3h 6h Custom

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

[Run query](#) ✔ 1 traces retrieved

[▶ Query refiners](#)

Traces (1) [Add to dashboard](#)

This table shows the most recent traces with an average response time of 2.11s. It shows as many as 1000 traces.

< 1 > [⚙](#)

ID	Trace status	Timestamp	Response code	Response Time	Duration
...86b347fc50bc57a992e9b835	✔ OK	19.1min (2022-08-10 12:05:25)	-	2.11s	2.11s

4. Pilih subsegmen Amazon SNS di bagian bawah timeline segmen, dan pilih tab Pengecualian untuk subsegmen SNS untuk melihat detail pengecualian.

Segments Timeline [Info](#)

	Segment status	Response code	Duration	
▼ Scorekeep AWS::EC2::Instance				0.0ms 200ms 400ms 600ms 800ms 1.0s 1.2s 1.4s 1.6s 1.8s 2.0s 2.2s
Scorekeep	✔ OK	-	2.11s	
SNS	✘ Fault (5xx)	400	728ms	Subscribe
▼ SNS AWS::SNS				
SNS	⚠ Error (4xx)	400	728ms	Subscribe

Segment details: SNS

[Overview](#) [Resources](#) [Exceptions](#)

Exceptions

Working Directory	Paths	message
-	-	Invalid parameter: Email address (Service: AmazonSNS; Status Code: 400; Error Code: InvalidParameter; Request ID: 8b80c997-630d-5c94-a67f-92f960ba0d3e)

Penyebabnya menunjukkan bahwa alamat email yang diberikan dalam panggilan ke `createSubscription` yang dibuat di kelas `WebConfig` tidak valid. Di bagian selanjutnya, kita akan memperbaikinya.

Mengonfigurasi notifikasi Amazon SNS

Scorekeep menggunakan Amazon SNS untuk mengirim notifikasi saat pengguna menyelesaikan game. Saat aplikasi dijalankan, ia mencoba membuat langganan untuk alamat email yang ditentukan dalam parameter CloudFormation tumpukan. Panggilan itu saat ini gagal. Konfigurasi email notifikasi untuk mengaktifkan notifikasi, dan selesaikan kegagalan yang disorot di peta jejak.

AWS Management Console

Untuk mengonfigurasi notifikasi Amazon SNS menggunakan AWS Management Console

1. Buka [konsol CloudFormation](#)
2. Pilih tombol radio di sebelah nama `scorekeep` tumpukan dalam daftar, lalu pilih Perbarui.
3. Pastikan bahwa `Use current template` dipilih, lalu klik `Next` pada halaman `Update stack`.
4. Temukan parameter `Email` dalam daftar, dan ganti nilai default dengan alamat email yang valid.

EcsInstanceTypeT3

Specifies the EC2 instance type for your container instances. Defaults to t3.micro.

t3.micro

Email

UPDATE_ME@

FrontendImageUri

public.ecr.aws/xray/scorekeep-frontend:latest

5. Gulir ke bagian bawah halaman dan pilih `Berikutnya`.
6. Gulir ke bagian bawah halaman `Tinjauan`, pilih kotak centang yang mengakui yang CloudFormation dapat membuat sumber daya IAM dengan nama khusus, dan pilih `Perbarui tumpukan`.
7. CloudFormation Tumpukan sekarang sedang diperbarui. Status tumpukan akan `UPDATE_IN_PROGRESS` sekitar lima menit sebelum diubah menjadi `UPDATE_COMPLETE`. Status akan disegarkan secara berkala, atau Anda dapat me-refresh halaman.

AWS CLI

Untuk mengonfigurasi notifikasi Amazon SNS menggunakan AWS CLI

1. Arahkan ke `xray-scorekeep/cloudformation/` folder yang sebelumnya Anda buat, dan buka `cf-resources.yaml` file di editor teks.
2. Temukan Default nilai dalam parameter Email dan ubah dari **UPDATE_ME** ke alamat email yang valid.

```
Parameters:
  Email:
    Type: String
    Default: UPDATE_ME # <- change to a valid abc@def.xyz email address
```

3. Dari `cloudformation` folder, perbarui CloudFormation tumpukan dengan AWS CLI perintah berikut:

```
aws cloudformation update-stack --stack-name scorekeep --capabilities
"CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

4. Tunggu sampai status CloudFormation tumpukan `UPDATE_COMPLETE`, yang akan memakan waktu beberapa menit. Gunakan AWS CLI perintah berikut untuk memeriksa status:

```
aws cloudformation describe-stacks --stack-name scorekeep --query
"Stacks[0].StackStatus"
```

Ketika pembaruan selesai, Scorekeep memulai ulang dan membuat langganan topik SNS. Periksa email Anda dan konfirmasikan langganan untuk melihat pembaruan saat Anda menyelesaikan game. Buka peta jejak untuk memverifikasi bahwa panggilan ke SNS tidak lagi gagal.

Jelajahi aplikasi sampel

Aplikasi sampel adalah API web HTTP di Java yang dikonfigurasi untuk menggunakan X-Ray SDK for Java. Ketika Anda menerapkan aplikasi dengan CloudFormation template, itu akan membuat tabel DynamoDB, Amazon ECS Cluster, dan layanan lain yang diperlukan untuk menjalankan Scorekeep di ECS. File definisi tugas untuk ECS dibuat melalui CloudFormation. File ini mendefinisikan gambar kontainer yang digunakan per tugas dalam cluster ECS. Gambar-gambar ini diperoleh dari ECR publik X-Ray resmi. Image container API scorekeep memiliki API yang dikompilasi dengan Gradle.

Gambar kontainer dari wadah frontend Scorekeep melayani frontend menggunakan server proxy nginx. Server ini merutekan permintaan ke jalur yang dimulai dengan /api ke API.

Untuk instrumentasi permintaan HTTP masuk, aplikasi menambahkan `TracingFilter` yang disediakan oleh SDK.

Example `src/main/java/scorekeep/WebConfig.java` - filter servlet

```
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
...

@Configuration
public class WebConfig {

    @Bean
    public Filter TracingFilter() {
        return new AWSXRayServletFilter("Scorekeep");
    }
    ...
}
```

Filter ini mengirimkan pelacakan data tentang semua permintaan masuk yang melayani aplikasi, termasuk permintaan URL, metode, status respons, waktu mulai, dan waktu berakhir.

Aplikasi ini juga membuat panggilan hilir ke DynamoDB menggunakan AWS SDK untuk Java. Untuk instrumen panggilan ini, aplikasi hanya mengambil submodul AWS terkait SDK sebagai dependensi, dan X-Ray SDK for Java secara otomatis menginstrumentasikan semua klien SDK. AWS

Aplikasi ini menggunakan Docker untuk membangun kode sumber on-instance dengan Gradle Docker Image dan Scorekeep API Dockerfile file untuk menjalankan JAR yang dapat dieksekusi yang dihasilkan Gradle. ENTRYPOINT

Example penggunaan Docker untuk membangun melalui Gradle Docker Image

```
docker run --rm -v /PATH/TO/SCOREKEEP_REPO/home/gradle/project -w /home/gradle/project
gradle:4.3 gradle build
```

Example TITIK MASUK Dockerfile

```
ENTRYPOINT [ "sh", "-c", "java -Dserver.port=5000 -jar scorekeep-api-1.0.0.jar" ]
```


File `build.gradle` mengunduh submodul SDK dari Maven selama kompilasi dengan menyatakannya sebagai dependensi.

Example `build.gradle` -- dependensi

```
...
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")
    }
}
```

Submodul Instrumentor inti, AWS AWS SDK, dan SDK adalah semua yang diperlukan untuk secara otomatis instrumen panggilan hilir yang dibuat dengan SDK. AWS

Untuk menyampaikan data segmen mentah ke X-Ray API, daemon X-Ray diperlukan untuk mendengarkan lalu lintas pada port UDP 2000. Untuk melakukannya, aplikasi memiliki daemon X-Ray yang dijalankan dalam wadah yang digunakan bersama aplikasi Scorekeep pada ECS sebagai wadah sespan. Lihat topik [daemon X-Ray](#) untuk informasi lebih lanjut.

Example Definisi Kontainer Daemon X-Ray dalam Definisi Tugas 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
...

```

X-Ray SDK for Java menyediakan kelas bernama `AWSXRay` yang menyediakan pencatat global, sebuah `TracingHandler` yang dapat Anda gunakan untuk instrumentasi kode Anda. Anda dapat mengonfigurasi pencatat global untuk menyesuaikan `AWSXRayServletFilter` yang membuat segmen untuk panggilan HTTP masuk. Sampel mencakup blok statis di kelas `WebConfig` yang mengonfigurasi pencatat global dengan plugin dan aturan sampling.

Example `src/main/java/scorekeep/WebConfig.java` - perekam

```

import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.plugins.ECSPPlugin;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;
...

@Configuration
public class WebConfig {
    ...

    static {
        AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
        ECSPPlugin()).withPlugin(new EC2Plugin());

        URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
        builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

        AWSXRay.setGlobalRecorder(builder.build());
        ...
    }
}

```

Contoh ini menggunakan builder untuk memuat aturan sampling dari file bernama `sampling-rules.json`. Aturan sampling menentukan tingkat saat SDK mencatat segmen untuk permintaan masuk.

Example `src/main/java/resources/sampling-aturan.json`

```
{
  "version": 1,
  "rules": [
    {
      "description": "Resource creation.",
      "service_name": "*",
      "http_method": "POST",
      "url_path": "/api/*",
      "fixed_target": 1,
      "rate": 1.0
    },
    {
      "description": "Session polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/session/*",
      "fixed_target": 0,
      "rate": 0.05
    },
    {
      "description": "Game polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/game/*/\"",
      "fixed_target": 0,
      "rate": 0.05
    },
    {
      "description": "State polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/state/*/\"",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
```

```
    "fixed_target": 1,  
    "rate": 0.1  
  }  
}
```

File aturan sampling menentukan empat aturan sampling kustom dan aturan default. Untuk setiap permintaan masuk, SDK mengevaluasi aturan kustom sesuai urutan penentuannya. SDK menerapkan aturan pertama yang cocok dengan permintaan metode, jalur, dan nama layanan. Untuk Scorekeep, aturan pertama menangkap semua permintaan POST (panggilan pembuatan sumber daya) dengan menerapkan target tetap dari satu permintaan per detik dan tingkat 1,0, atau 100 persen permintaan setelah target tetap terpenuhi.

Tiga aturan kustom lainnya menerapkan tingkat lima persen tanpa target tetap untuk sesi, game, dan baca status (permintaan GET). Hal ini meminimalkan jumlah pelacakan untuk panggilan periodik yang dibuat front end secara otomatis setiap beberapa detik untuk memastikan konten diperbarui. Untuk semua permintaan lainnya, file menentukan tingkat default dari satu permintaan per detik dan tingkat 10 persen.

Aplikasi sampel juga menunjukkan cara menggunakan fitur lanjutan seperti instrumentasi klien SDK manual, membuat tambahan subsegmen, dan panggilan HTTP keluar. Untuk informasi selengkapnya, lihat [AWS X-Ray aplikasi sampel](#).

Opsional: Kebijakan hak istimewa terbatas

Kontainer Scorekeep ECS mengakses sumber daya menggunakan kebijakan akses penuh, seperti `AmazonSNSFullAccess` `AmazonDynamoDBFullAccess`. Menggunakan kebijakan akses penuh bukanlah praktik terbaik untuk aplikasi produksi. Contoh berikut memperbarui kebijakan DynamoDB IAM untuk meningkatkan keamanan aplikasi. Untuk mempelajari lebih lanjut tentang praktik terbaik keamanan dalam kebijakan IAM, lihat [Identitas dan manajemen akses untuk AWS X-Ray](#).

Example template `cf-resources.yaml` Definisi peran `ECSTask`

```
ECSTaskRole:  
  Type: AWS::IAM::Role  
  Properties:  
    AssumeRolePolicyDocument:  
      Version: "2012-10-17"  
      Statement:  
        -
```

```

    Effect: "Allow"
    Principal:
      Service:
        - "ecs-tasks.amazonaws.com"
    Action:
      - "sts:AssumeRole"
    ManagedPolicyArns:
      - "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
      - "arn:aws:iam::aws:policy/AmazonSNSFullAccess"
      - "arn:aws:iam::aws:policy/AWSXrayFullAccess"
    RoleName: "scorekeepRole"

```

Untuk memperbarui kebijakan Anda, pertama-tama Anda mengidentifikasi ARN sumber daya DynamoDB Anda. Kemudian Anda menggunakan ARN dalam kebijakan IAM khusus. Terakhir, Anda menerapkan kebijakan itu ke profil instans Anda.

Untuk mengidentifikasi ARN sumber daya DynamoDB Anda:

1. Buka [Konsol DynamoDB](#).
2. Pilih Tabel dari bilah navigasi kiri.
3. Pilih salah satu scorekeep-* untuk menampilkan halaman detail tabel.
4. Di bawah tab Ikhtisar, pilih Info tambahan untuk memperluas bagian dan melihat Nama Sumber Daya Amazon (ARN). Salin nilai ini.
5. Masukkan ARN ke dalam kebijakan IAM berikut, ganti AWS_ACCOUNT_ID nilai AWS_REGION dan dengan wilayah dan ID akun spesifik Anda. Kebijakan baru ini hanya mengizinkan tindakan yang ditentukan, bukan AmazonDynamoDBFullAccess kebijakan yang mengizinkan tindakan apa pun.

Example

```

{
  "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:<AWS_REGION>:<AWS_ACCOUNT_ID>:table/
scorekeep-*"
    }
  ]
}
```

Tabel yang dibuat aplikasi mengikuti konvensi penamaan yang konsisten. Anda dapat menggunakan `scorekeep-*` format untuk menunjukkan semua tabel Scorekeep.

Ubah kebijakan IAM Anda

1. Buka [peran tugas Scorekeep \(ScoreKeeProle\)](#) dari konsol IAM.
2. Pilih kotak centang di samping `AmazonDynamoDBFullAccess` kebijakan dan pilih Hapus untuk menghapus kebijakan ini.
3. Pilih Tambahkan izin, lalu Lampirkan kebijakan, dan terakhir Buat kebijakan.
4. Pilih tab JSON dan tempel kebijakan yang dibuat di atas.
5. Pilih Berikutnya: Tag di bagian bawah halaman.
6. Pilih Berikutnya: Tinjau di bagian bawah halaman.
7. Untuk Nama, tetapkan nama untuk kebijakan tersebut.
8. Pilih Buat kebijakan di bagian bawah halaman.
9. Lampirkan kebijakan yang baru dibuat ke `scorekeepRole` peran. Mungkin perlu beberapa menit agar kebijakan terlampir berlaku.

Jika Anda telah melampirkan kebijakan baru ke `scorekeepRole` peran, Anda harus melepaskannya sebelum menghapus CloudFormation tumpukan, karena kebijakan terlampir ini akan memblokir tumpukan agar tidak dihapus. Kebijakan dapat dipisahkan secara otomatis dengan menghapus kebijakan.

Hapus kebijakan IAM kustom Anda

1. Buka [konsol IAM](#).
2. Pilih Kebijakan dari bilah navigasi kiri.

3. Cari nama kebijakan kustom yang Anda buat sebelumnya di bagian ini, dan pilih tombol radio di sebelah nama kebijakan untuk menyorotnya.
4. Pilih drop-down Tindakan dan kemudian pilih Hapus.
5. Ketik nama kebijakan kustom, lalu pilih Hapus untuk mengonfirmasi penghapusan. Ini akan secara otomatis melepaskan kebijakan dari `scorekeepRole` peran.

Bersihkan

Ikuti langkah-langkah berikut untuk menghapus sumber daya aplikasi Scorekeep:

Note

Jika Anda membuat dan melampirkan kebijakan kustom menggunakan bagian sebelumnya dari tutorial ini, Anda harus menghapus kebijakan dari `scorekeepRole` sebelum menghapus CloudFormation tumpukan.

AWS Management Console

Hapus aplikasi sampel menggunakan AWS Management Console

1. Buka [konsol CloudFormation](#)
2. Pilih tombol radio di sebelah nama `scorekeep` tumpukan dalam daftar, lalu pilih Hapus.
3. CloudFormation Tumpukan sekarang sedang dihapus. Status tumpukan akan berlangsung `DELETE_IN_PROGRESS` selama beberapa menit sampai semua sumber daya dihapus. Status akan disegarkan secara berkala, atau Anda dapat me-refresh halaman.

AWS CLI

Hapus aplikasi sampel menggunakan AWS CLI

1. Masukkan AWS CLI perintah berikut untuk menghapus CloudFormation tumpukan:

```
aws cloudformation delete-stack --stack-name scorekeep
```

2. Tunggu sampai CloudFormation tumpukan tidak ada lagi, yang akan memakan waktu sekitar lima menit. Gunakan AWS CLI perintah berikut untuk memeriksa status:

```
aws cloudformation describe-stacks --stack-name scorekeep --query  
"Stacks[0].StackStatus"
```

Langkah selanjutnya

Pelajari selengkapnya tentang X-Ray di bab berikutnya, [AWS X-Ray konsep](#).

Untuk menggunakan aplikasi Anda sendiri, pelajari lebih lanjut tentang X-Ray SDK for Java atau salah satu SDKs X-Ray lainnya:

- X-Ray SDK for Java – [AWS X-Ray SDK for Java](#)
- X-Ray SDK for Node.js – [AWS X-Ray SDK untuk Node.js](#)
- X-Ray SDK for .NET – [AWS X-Ray SDK for .NET](#)

Untuk menjalankan daemon X-Ray secara lokal atau aktif, lihat. AWS [AWS X-Ray daemon](#)

Untuk berkontribusi pada aplikasi sampel GitHub, lihat [eb-java-scorekeep](#).

Menginstrumentasi klien AWS SDK secara manual

X-Ray SDK for Java secara otomatis menginstrumentasikan AWS semua klien SDK saat [Anda menyertakan submodul SDK Instrumentor AWS dalam](#) dependensi build Anda.

Anda dapat menonaktifkan instrumentasi klien otomatis dengan menghapus submodul Instrumentor. Hal ini memungkinkan Anda untuk menginstruksikan beberapa klien secara manual sambil mengabaikan yang lain, atau menggunakan pengendali pelacakan yang berbeda pada klien yang berbeda.

Untuk mengilustrasikan dukungan untuk menginstrumentasi klien AWS SDK tertentu, aplikasi meneruskan penanganan penelusuran `AmazonDynamoDBClientBuilder` sebagai penanganan permintaan dalam model pengguna, game, dan sesi. Perubahan kode ini memberi tahu SDK untuk memasukkan semua panggilan ke DynamoDB menggunakan klien tersebut.

Example [src/main/java/scorekeep/SessionModel.java](#) – Instrumentasi klien AWS SDK manual

```
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);
```

Jika Anda menghapus submodul AWS SDK Instrumentor dari dependensi proyek, hanya klien AWS SDK yang diinstrumentasi secara manual yang muncul di peta jejak.

Membuat subsegmen tambahan

Di kelas model pengguna, aplikasi secara manual membuat subsegmen untuk mengelompokkan semua panggilan hilir yang dibuat dalam fungsi `saveUser` dan menambahkan metadata.

Example [src/main/java/scorekeep/UserModel.java](#) - Subsegmen kustom

```
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();
    }
}
```

Merekam anotasi, metadata, dan pengguna IDs

Di kelas model game, aplikasi mencatat objek Game di blok [metadata](#) setiap kali game disimpan di DynamoDB. Secara terpisah, aplikasi merekam game IDs dalam [anotasi](#) untuk digunakan dengan [ekspresi filter](#).

Example [src/main/java/scorekeep/GameModel.java](#) – Anotasi dan metadata

```
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();
    }
}
```

Dalam pengontrol bergerak, aplikasi merekam [pengguna IDs](#) dengan `setUser`. Pengguna IDs direkam dalam bidang terpisah pada segmen dan diindeks untuk digunakan dengan pencarian.

Example [src/main/java/scorekeep/MoveController.java](#) — ID Pengguna

```
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);
}
```

Instrumentasi panggilan HTTP keluar

Kelas pabrik pengguna menunjukkan bagaimana aplikasi menggunakan versi X-Ray SDK for Java dari `HttpClientBuilder` untuk menginstrumen panggilan HTTP keluar.

Example [src/main/java/scorekeep/UserFactory.java](#)— `HttpClient` instrumentasi

```
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();
    }
}
```

Jika saat ini Anda menggunakan `org.apache.http.impl.client.HttpClientBuilder`, Anda cukup menukar pernyataan impor untuk kelas tersebut dengan satu untuk `com.amazonaws.xray.proxies.apache.http.HttpClientBuilder`.

Menginstrumentasi panggilan ke basis data PostgreSQL

`application-pgsql.properties` file menambahkan pencegat penelusuran X-Ray PostgreSQL ke sumber data yang dibuat di [RdsWebConfig.java](#).

Example [application-pgsql.properties](#) – Instrumentasi basis data 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

Lihat [Mengonfigurasi basis data dengan Elastic Beanstalk](#) di AWS Elastic Beanstalk Panduan Developer untuk detail tentang cara menambahkan basis data PostgreSQL ke lingkungan aplikasi.

Versi demo X-Ray di cabang `xray` termasuk demo yang menggunakan sumber data berinstrumen untuk menghasilkan pelacakan yang menunjukkan informasi tentang kueri SQL yang dihasilkannya. Navigasikan ke jalur `/#/xray` dalam aplikasi yang sedang berjalan atau pilih Didukung oleh AWS X-Ray di bilah navigasi untuk melihat halaman 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.

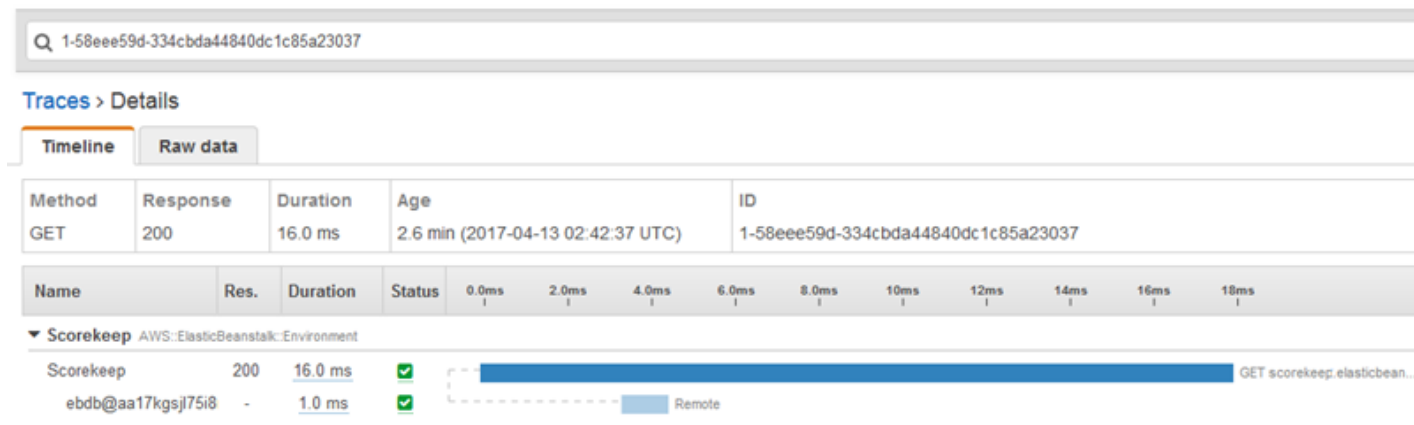
Trace SQL queries

[View traces in AWS X-Ray](#)

ID	Winner	Loser
1	Mugur	Gheorghită
2	Paula	Adorján
3	Αρχίας	Stela
4	付	Pervanə

Pilih Lacak kueri SQL untuk menyimulasikan sesi game dan menyimpan hasil dalam basis data terlampir. Kemudian, pilih Lihat jejak di AWS X-Ray untuk melihat daftar jejak yang difilter yang mengenai `/api/history` rute API.

Pilih salah satu pelacakan dari daftar untuk melihat linimasa, termasuk kueri SQL.



Fungsi instrumentasi AWS Lambda

Scorekeep menggunakan dua AWS Lambda fungsi. Yang pertama adalah fungsi Node.js dari cabang `lambda` yang menghasilkan nama acak untuk pengguna baru. Ketika pengguna membuat sesi tanpa memasukkan nama, aplikasi memanggil fungsi bernama `random-name` dengan AWS SDK untuk Java. X-Ray SDK for Java merekam informasi tentang panggilan ke Lambda di subsegmen seperti panggilan lain yang dibuat dengan klien SDK berinstrumen. AWS

Note

Menjalankan fungsi Lambda `random-name` membutuhkan pembuatan sumber daya tambahan di luar lingkungan Elastic Beanstalk. Lihat [readme](#) untuk informasi selengkapnya dan petunjuk: [Integrasi AWS Lambda](#).

Fungsi kedua, `scorekeep-worker`, adalah fungsi Python yang berjalan secara independen dari API Scorekeep. Saat game berakhir, API menulis ID sesi dan ID game ke antrean SQS. Fungsi pekerja membaca item dari antrean, dan memanggil API Scorekeep untuk membuat catatan lengkap dari setiap sesi game untuk penyimpanan di Amazon S3.

Scorekeep mencakup AWS CloudFormation template dan skrip untuk membuat kedua fungsi. Karena Anda harus memaketkan X-Ray SDK dengan kode fungsi, templat membuat fungsi tanpa

kode apa pun. Ketika Anda men-deploy Scorekeep, file konfigurasi yang disertakan dalam folder `.ebextensions` membuat paket bundel sumber yang meliputi SDK, dan memperbarui kode fungsi dan konfigurasi dengan AWS Command Line Interface.

Fungsi

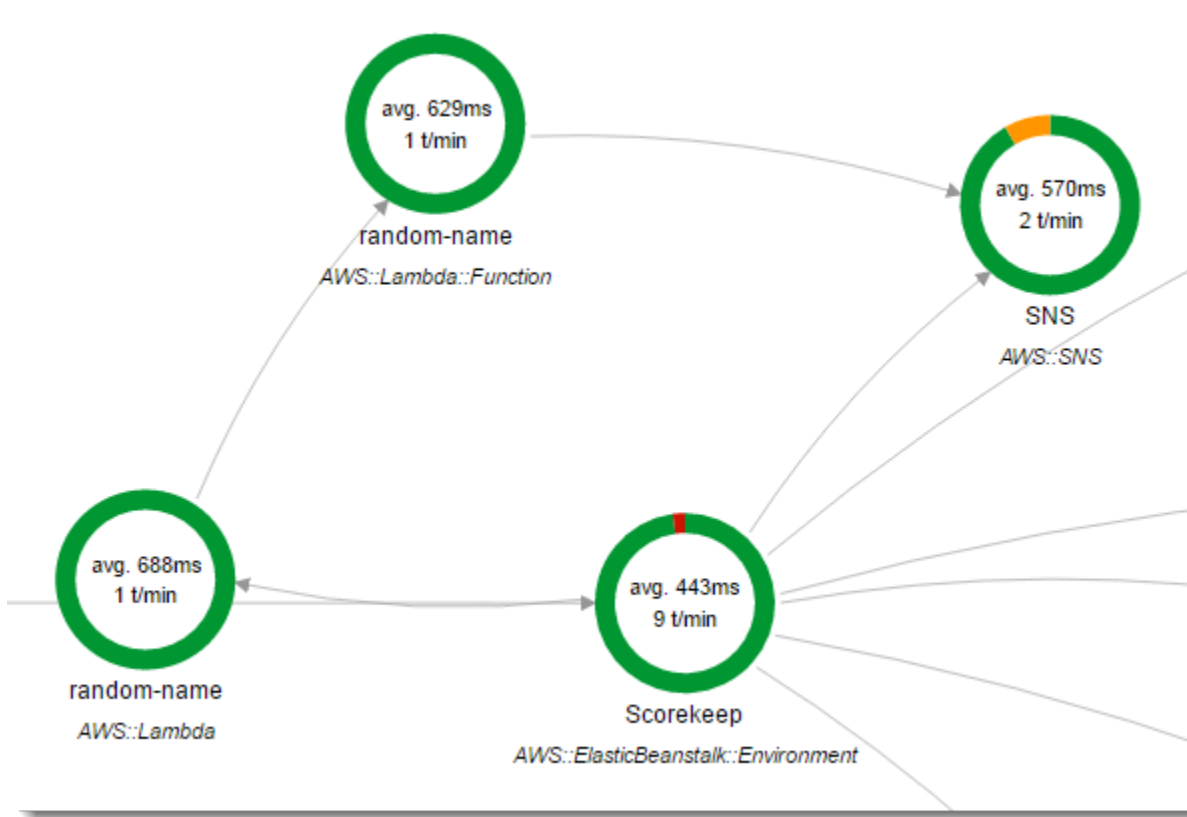
- [Nama acak](#)
- [Pekerja](#)

Nama acak

Scorekeep memanggil fungsi nama acak ketika pengguna memulai sesi game tanpa masuk atau menentukan nama pengguna. Saat Lambda memproses panggilan ke `random-name`, Lambda membaca [header pelacakan](#), yang berisi ID pelacakan dan keputusan pengambilan sampel yang ditulis oleh X-Ray SDK for Java.

Untuk setiap permintaan sampel, Lambda menjalankan daemon X-Ray dan menulis dua segmen. Segmen pertama mencatat informasi tentang panggilan ke Lambda yang memanggil fungsi. Segmen ini berisi informasi yang sama dengan subsegmen yang dicatat oleh Scorekeep, namun dari sudut pandang Lambda. Segmen kedua menunjukkan pekerjaan yang dilakukan fungsi.

Lambda melewati segmen fungsi ke X-Ray SDK melalui konteks fungsi. Saat Anda menginstrumentasi fungsi Lambda, Anda tidak boleh menggunakan SDK untuk [membuat segmen untuk permintaan masuk](#). Lambda menyediakan segmen tersebut, dan Anda menggunakan SDK untuk menginstrumentasi klien dan menulis subsegment.



Fungsi `random-name` diimplementasikan dalam Node.js. Ini menggunakan SDK untuk JavaScript di Node.js untuk mengirim notifikasi dengan Amazon SNS, dan X-Ray SDK untuk Node.js untuk instrumen AWS klien SDK. Untuk menulis anotasi, fungsi membuat subsegmen kustom dengan `AWSXRay.captureFunc`, dan menulis anotasi dalam fungsi yang diinstrumentasi. Di Lambda, Anda tidak dapat menulis anotasi langsung ke segmen fungsi, namun hanya ke subsegmen yang Anda buat.

Example [function/index.js](#) – Fungsi Lambda nama acak

```
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;

```

Fungsi ini dibuat secara otomatis ketika Anda men-deploy aplikasi sampel ke Elastic Beanstalk. Cabang xray termasuk skrip untuk membuat fungsi Lambda kosong. File konfigurasi dalam .ebextensions folder membangun paket fungsi dengan npm install selama penerapan, dan kemudian memperbarui fungsi Lambda dengan CLI. AWS

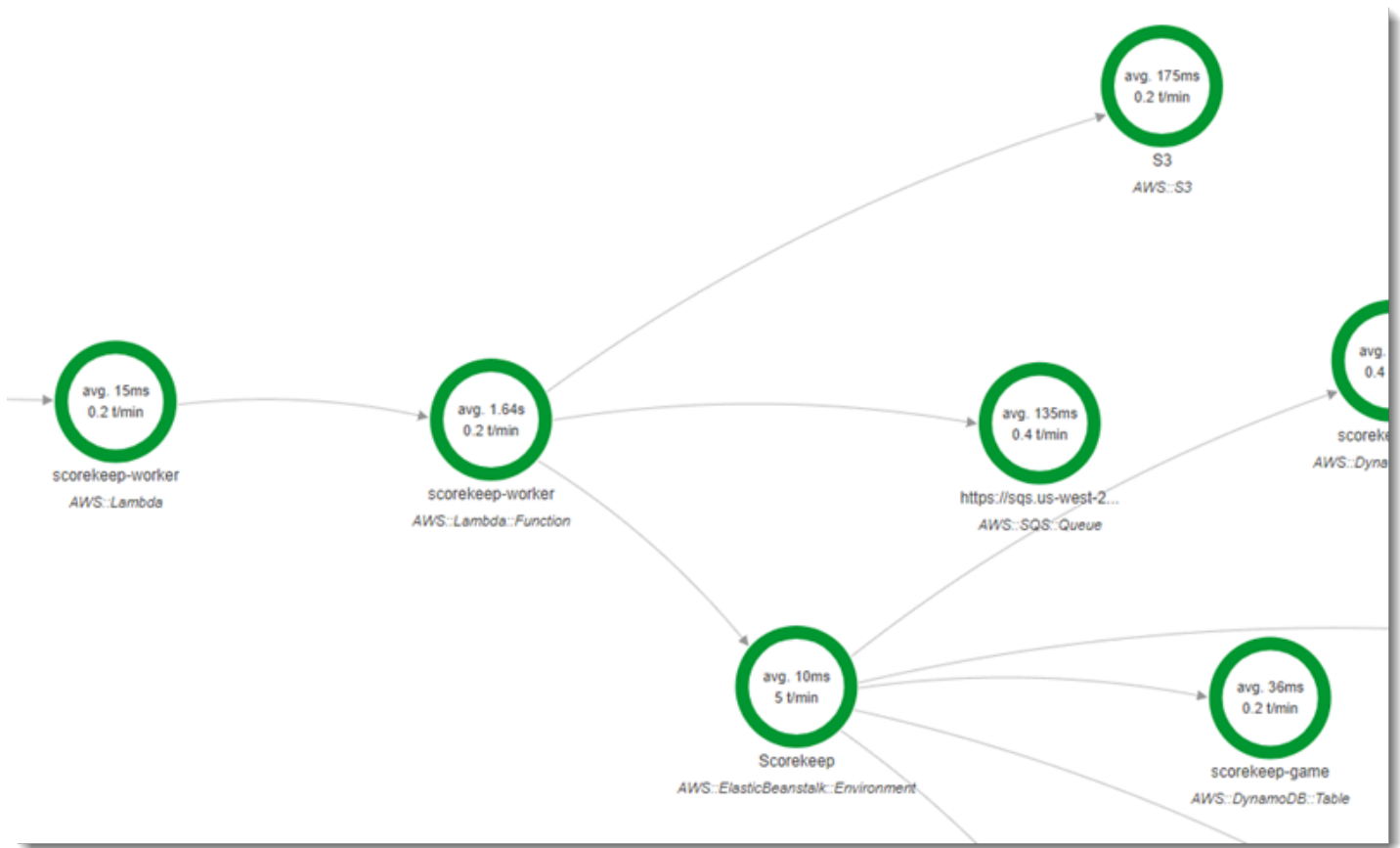
Pekerja

Fungsi pekerja yang diinstrumentasi disediakan di cabangnya sendiri, xray-worker, karena tidak dapat berjalan kecuali Anda membuat fungsi pekerja dan sumber daya terkait terlebih dahulu. Lihat [readme cabang](#) untuk instruksi.

Fungsi ini dipicu oleh CloudWatch acara Amazon Events yang dibundel setiap 5 menit. Ketika berjalan, fungsi menarik item dari antrean Amazon SQS yang dikelola Scorekeep. Setiap pesan berisi informasi tentang game yang sudah selesai.

Pekerja menarik catatan dan dokumen game dari tabel lain tempat game mencatat referensi.. Misalnya, catatan game di DynamoDB meliputi daftar gerakan yang dieksekusi selama game berlangsung. Daftar ini tidak berisi gerakan itu sendiri, melainkan IDs gerakan yang disimpan dalam tabel terpisah.

Sesi, dan status juga disimpan sebagai referensi. Hal ini membuat entri dalam tabel game menjadi terlalu besar, tetapi membutuhkan panggilan tambahan untuk mendapatkan semua informasi tentang game. Pekerja melakukan deferensi pada semua entri ini dan membuat catatan lengkap game sebagai dokumen tunggal di Amazon S3. Saat Anda ingin melakukan analitik pada data, Anda dapat melakukan kueri langsung di atasnya di Amazon S3 dengan Amazon Athena tanpa menjalankan migrasi data baca-berat untuk mendapatkan data Anda dari DynamoDB.



Fungsi pekerja juga mengaktifkan pelacakan aktif dalam konfigurasinya di AWS Lambda. Tidak seperti fungsi nama acak, pekerja tidak menerima permintaan dari aplikasi yang diinstrumentasi, sehingga AWS Lambda tidak menerima header penelusuran. Dengan pelacakan aktif, Lambda membuat ID pelacakan dan membuat keputusan pengambilan sampel.

X-Ray SDK untuk Python hanyalah beberapa baris di bagian atas fungsi yang mengimpor SDK dan menjalankan `patch_all` fungsinya untuk menambal AWS SDK for Python (Boto) dan yang

digunakannya untuk memanggil Amazon SQS HTTP clients dan Amazon S3. Ketika pekerja memanggil API Scorekeep, SDK menambahkan [header pelacakan](#) pada permintaan untuk melacak panggilan melalui API.

Example [_lambda/scorekeep-worker/scorekeep-worker.py](#) – Fungsi Lambda pekerja

```
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')

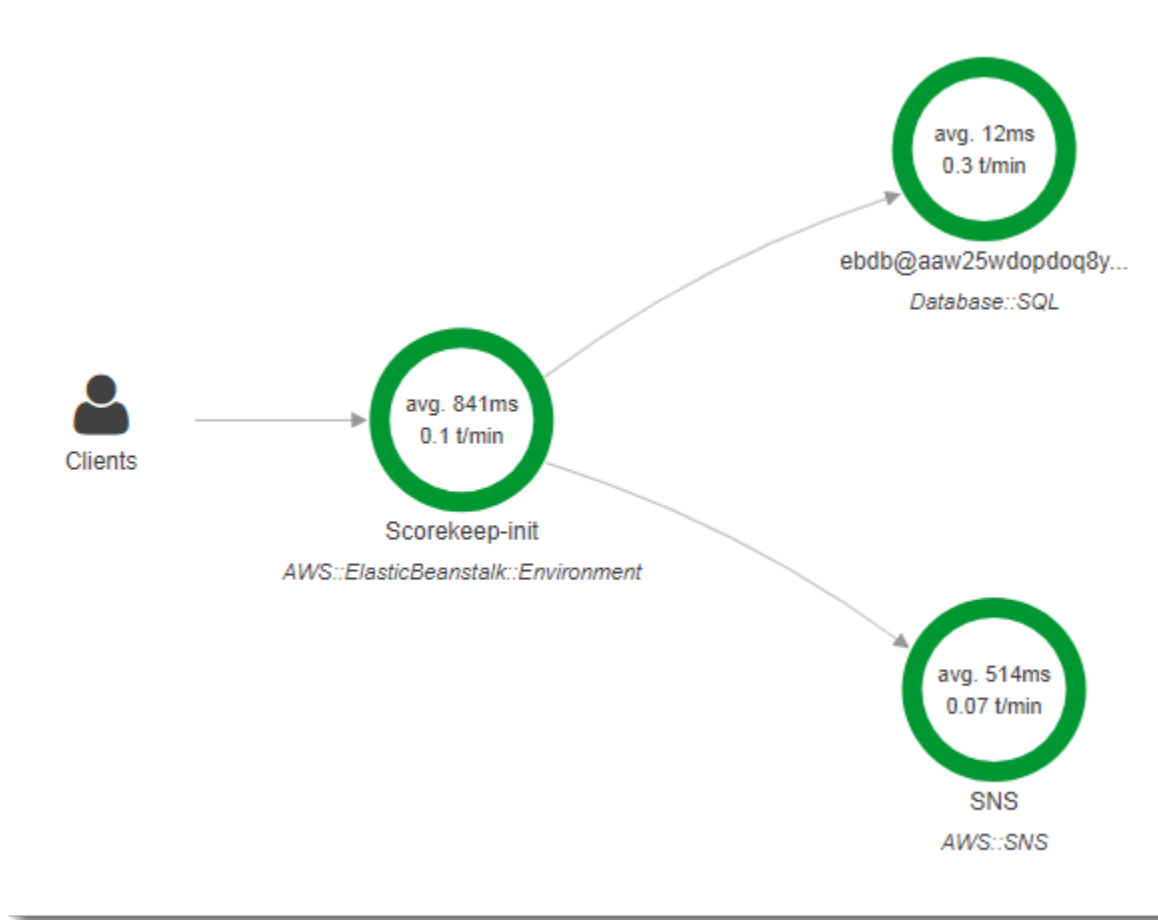
    # Receive message from SQS queue
    response = sqs.receive_message(
        QueueUrl=queue_url,
        AttributeNames=[
            'SentTimestamp'
        ],
        MaxNumberOfMessages=1,
        MessageAttributeNames=[
            'All'
        ],
        VisibilityTimeout=0,
        WaitTimeSeconds=0
    )
    ...
```

Menginstrumentasi kode perusahaan rintisan

X-Ray SDK for Java secara otomatis membuat segmen untuk permintaan masuk. Selama permintaan berada dalam cakupan, Anda dapat menggunakan klien berinstrumen dan mencatat subsegmen

tanpa masalah. Jika Anda mencoba menggunakan klien berinstrumen dalam kode startup, Anda akan mendapatkan [SegmentNotFoundException](#).

Kode perusahaan rintisan berjalan di luar aliran permintaan/respons standar aplikasi web, jadi Anda perlu membuat segmen secara manual untuk melengkapinya. Scorekeep menunjukkan instrumentasi kode perusahaan rintisan dalam file `WebConfig`. Scorekeep memanggil basis data SQL dan Amazon SNS selama perusahaan rintisan.



Kelas `WebConfig` default membuat langganan Amazon SNS untuk notifikasi. Untuk menyediakan segmen bagi SDK X-Ray untuk menulis ketika klien Amazon SNS digunakan, Scorekeep memanggil `beginSegment` dan `endSegment` pada catatan global.

Example [src/main/java/scorekeep/WebConfig.java](#) – Klien AWS yang diinstrumentasi dalam kode perusahaan rintisan

```

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();
```

Di `RdsWebConfig`, yang Scorekeep gunakan saat basis data Amazon RDS terhubung, konfigurasi juga membuat segmen untuk klien SQL yang digunakan Hibernate saat menerapkan skema basis data selama perusahaan rintisan.

Example [src/main/java/scorekeep/RdsWebConfig.java](#) – Klien basis data SQL yang diinstrumentasi dalam kode perusahaan rintisan

```
@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` berjalan secara otomatis dan menggunakan klien SQL. Karena klien diinstrumentasi, Scorekeep harus mengganti implementasi default dan menyediakan segmen SDK untuk digunakan ketika klien dipanggil.

Skrip instrumentasi

Anda juga dapat menginstrumentasi kode yang bukan bagian dari aplikasi Anda. Ketika daemon X-Ray berjalan, ia akan menyampaikan segmen apa pun yang diterimanya ke X-Ray, meskipun tidak dihasilkan oleh SDK X-Ray. Scorekeep menggunakan skrip sendiri untuk instrumen membangun yang mengompilasi aplikasi selama deployment.

Example [bin/build.sh](#) – Skrip bangunan berinstrumen

```
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](#) dan [xray_success.py](#) adalah skrip Python sederhana yang mengonstruksi objek segmen, mengonversinya menjadi dokumen JSON, dan mengirimkannya ke daemon melalui UDP. Jika build Gradle gagal, Anda dapat menemukan pesan kesalahan dengan mengklik node scorekeep-build di peta jejak konsol X-Ray.



Traces > Details

Timeline

Raw data

Method	Response	Duration	Age	ID
--	--	14.6 sec	4.5 min (2017-09-14 01:25:01 UTC)	1-59b9da6d-ab8ca2666217b31a03eff86d

Name	Res.	Duration	Status	0.0ms	2.0s	4.0s	6.0s	8.0s	10s	12s	14s	16s

▼ Scorekeep-build

Scorekeep-build

-

14.6 sec

--

Segment - Scorekeep-build

Overview

Resources

Annotations

Metadata

Exceptions

Working directory

/var/app/current

Paths

/var/app/current/src/main/java/scorekeep/

Cause

/var/app/staging/src/main/java/scorekeep/RdsWebConfig.java:89: error: cannot find symbol

AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new EC2Plugin()).withPlugin(new ElasticBeanstalkPlugin());

A

symbol: class ElasticBeanstalkPlugin

location: class RdsWebConfig

1 error

FAILURE: Build failed with an exception.

Menginstrumentasi klien aplikasi web

Di cabang [xray-cognito](#), Scorekeep menggunakan Amazon Cognito untuk memungkinkan pengguna membuat akun dan masuk dengan akun tersebut untuk mengambil informasi pengguna mereka dari kolam pengguna Amazon Cognito. Saat pengguna masuk, Scorekeep menggunakan kumpulan identitas Amazon Cognito untuk mendapatkan kredensi AWS sementara untuk digunakan dengan AWS SDK untuk JavaScript

Kolam identitas dikonfigurasi untuk membiarkan masuk pengguna menulis penelusuran data ke AWS X-Ray. Aplikasi web menggunakan kredensialnya untuk mencatat ID pengguna yang masuk, jalur peramban, dan tampilan panggilan klien ke API Skorrekeep.

Sebagian besar pekerjaan dilakukan di kelas layanan bernama xray. Kelas layanan ini menyediakan metode untuk menghasilkan pengidentifikasi yang diperlukan, menciptakan segmen yang sedang berlangsung, menyelesaikan segmen, dan mengirimkan dokumen segmen ke API X-Ray.

Example **public/xray.js** – Catat dan unggah segmen

```
...
    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);  
    }  
  })  
}
```



```

    })
  }
}

```

Metode ini disebut di header dan fungsi `transformResponse` di layanan sumber daya yang digunakan aplikasi web untuk memanggil API Skorrekeep. Untuk menyertakan segmen klien dalam pelacakan yang sama sebagai segmen yang menghasilkan API, aplikasi web harus menyertakan ID pelacakan dan ID segmen di [header pelacakan](#) (`X-Amzn-Trace-Id`) yang dapat dibaca X-Ray SDK. Ketika aplikasi Java instrumented menerima permintaan dengan header ini, X-Ray SDK for Java menggunakan ID penelusuran yang sama dan membuat segmen dari klien aplikasi web induk dari segmennya.

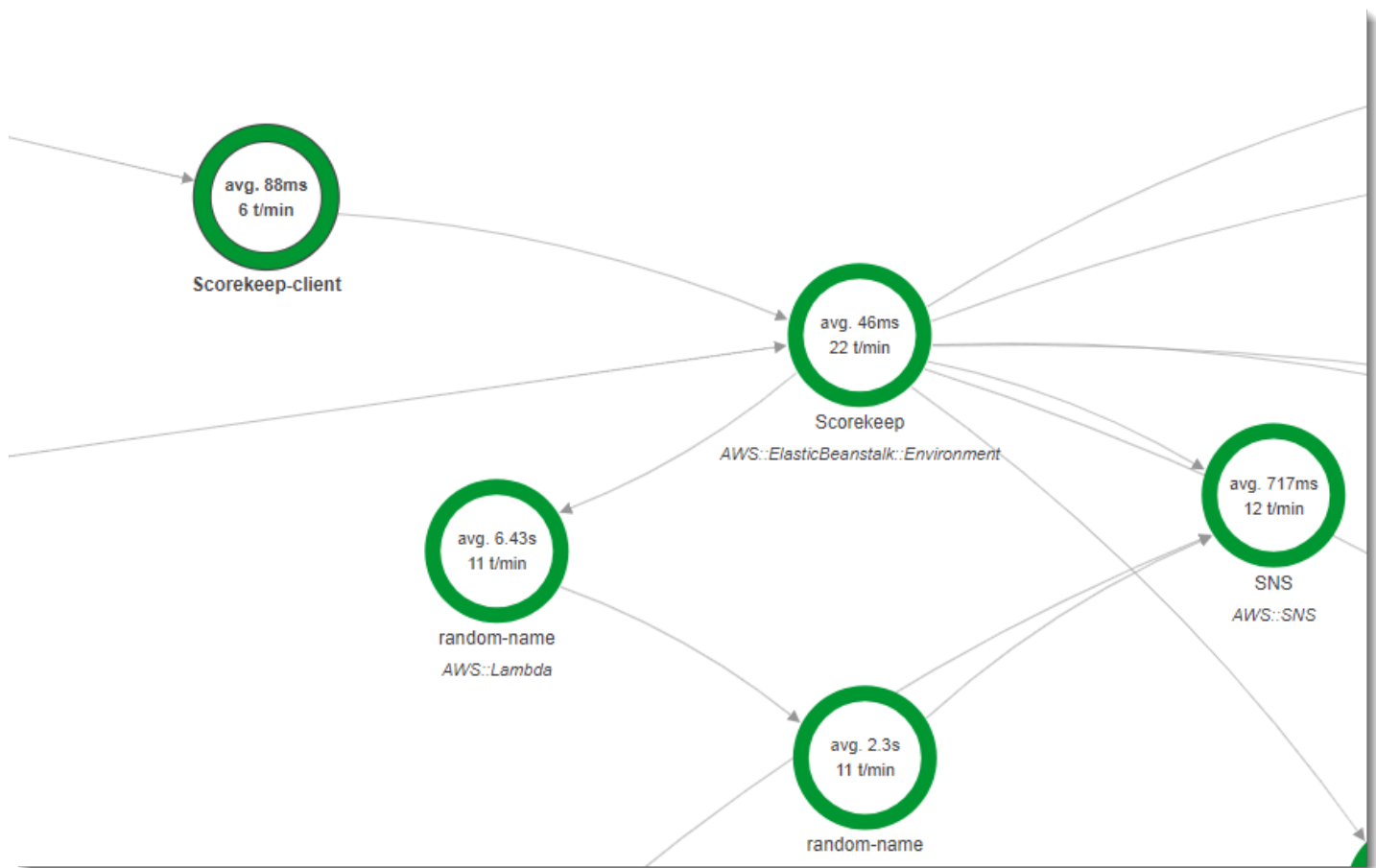
Example [public/app/services.js](#) - Mencatat segmen untuk panggilan sumber daya sudut dan menulis header pelacakan

```

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);
    },
  },
  ...

```

Peta jejak yang dihasilkan menyertakan node untuk klien aplikasi web.



Penelusuran yang mencakup segmen dari aplikasi web menampilkan URL yang dilihat pengguna di peramban (jalur yang dimulai dengan `/#/`). Tanpa instrumentasi klien, Anda hanya mendapatkan URL sumber daya API yang dipanggil aplikasi web (path dimulai dengan `/api/`).

Trace overview

Group by: URL

URL	Avg response time
http://scorekeep.elasticbeanstalk.com/#/	86.2 ms
http://scorekeep.elasticbeanstalk.com/#/session/4ORP7OB5/47H4SETD	58.5 ms
http://scorekeep.elasticbeanstalk.com/#/game/4ORP7OB5/A94SAFFD/47H4SETD	255 ms

Menggunakan klien berinstrumen di utas pekerja

Scorekeep menggunakan utas pekerja untuk memublikasikan notifikasi ke Amazon SNS ketika pengguna memenangkan game. Penerbitan notifikasi membutuhkan waktu lebih lama daripada gabungan operasi permintaan lainnya, dan tidak memengaruhi klien atau pengguna. Oleh karena itu, melakukan tugas secara asinkron adalah cara yang baik untuk meningkatkan waktu respons.

Namun, X-Ray SDK for Java tidak mengetahui segmen mana yang aktif saat utas dibuat. Akibatnya, ketika Anda mencoba menggunakan AWS SDK untuk Java klien yang diinstrumentasi di dalam utas, itu akan melempar utas yang `SegmentNotFoundException` rusak.

Example Web-1.error.log

```
Exception in thread "Thread-2" com.amazonaws.xray.exceptions.SegmentNotFoundException:
  Failed to begin subsegment named 'AmazonSNS': segment cannot be found.
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at
  sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
    at
  sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:
  ...
```

Untuk memperbaikinya, aplikasi menggunakan `GetTraceEntity` untuk mendapatkan referensi ke segmen di utas utama, dan `Entity.run()` untuk menjalankan kode utas pekerja dengan aman dengan akses ke konteks segmen.

Example [src/main/java/scorekeep/MoveFactory.java](#) – Melewati konteks pelacakan ke utas pekerja

```
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;
...

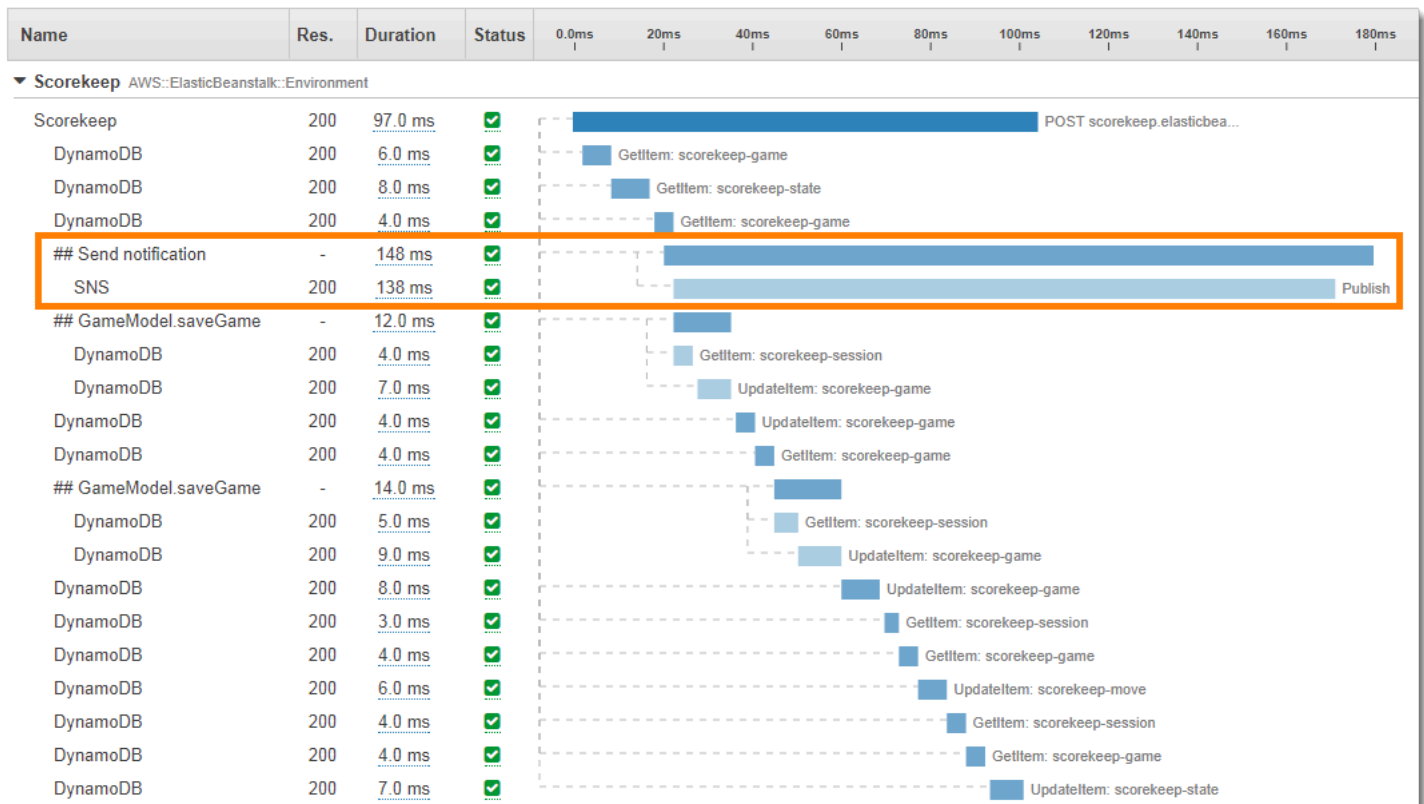
Entity segment = recorder.getTraceEntity();
Thread comm = new Thread() {
    public void run() {
        segment.run(() -> {
            Subsegment subsegment = AWSXRay.beginSubsegment("## Send notification");
            Sns.sendNotification("Scorekeep game completed", "Winner: " + userId);
        });
    }
};
```

```

    AWSXRay.endSubsegment();
  }
}

```

Karena permintaan sekarang diselesaikan sebelum panggilan ke Amazon SNS, aplikasi membuat subsegmen terpisah untuk utas. Hal ini mencegah X-Ray SDK menutup segmen sebelum mencatat respons dari Amazon SNS. Jika tidak ada subsegmen yang terbuka saat Scorekeep menyelesaikan permintaan, respons dari Amazon SNS bisa hilang.



Lihat [Melewati konteks segmen antara benang dalam aplikasi multithreaded](#) untuk informasi selengkapnya tentang multithreading.

AWS X-Ray daemon

Note

Anda sekarang dapat menggunakan CloudWatch agen untuk mengumpulkan metrik, log, dan jejak dari EC2 instans Amazon dan server di lokasi. CloudWatch agen versi 1.300025.0 dan yang lebih baru dapat mengumpulkan jejak dari atau klien [OpenTelemetry](#) X-Ray SDKs, dan mengirimkannya ke [X-Ray](#). Menggunakan CloudWatch agen alih-alih AWS Distro for OpenTelemetry (ADOT) Collector atau daemon X-Ray untuk mengumpulkan jejak dapat membantu Anda mengurangi jumlah agen yang Anda kelola. Lihat topik [CloudWatch agen](#) di Panduan CloudWatch Pengguna untuk informasi selengkapnya.

AWS X-Ray Daemon adalah aplikasi perangkat lunak yang mendengarkan lalu lintas pada port UDP 2000, mengumpulkan data segmen mentah, dan menyampaikannya ke API. AWS X-Ray Daemon bekerja bersama dengan AWS X-Ray SDKs dan harus berjalan sehingga data yang dikirim oleh SDKs dapat mencapai layanan X-Ray. X-Ray Daemon adalah proyek sumber terbuka. Anda dapat mengikuti proyek dan mengirimkan masalah dan menarik permintaan di GitHub: github.com/aws/aws-xray-daemon

AWS Lambda Aktif dan AWS Elastic Beanstalk, gunakan integrasi layanan tersebut dengan X-Ray untuk menjalankan daemon. Lambda menjalankan daemon secara otomatis setiap kali suatu fungsi dipanggil untuk permintaan sampel. Pada Elastic Beanstalk, [gunakan XRayEnabled opsi konfigurasi](#) untuk menjalankan daemon pada instans di lingkungan Anda. Untuk informasi selengkapnya, silakan lihat

Untuk menjalankan daemon X-Ray secara lokal, lokal, atau lainnya Layanan AWS, unduh, [jalankan](#), lalu [berikan izin](#) untuk mengunggah dokumen segmen ke X-Ray.

Mengunduh daemon

Anda dapat mengunduh daemon dari Amazon S3, Amazon ECR, atau Docker Hub, lalu menjalankannya secara lokal, atau menginstalnya di instans Amazon saat diluncurkan. EC2

Amazon S3

Pemasang dan executable X-Ray daemon

- Linux (dapat dieksekusi) – [aws-xray-daemon-linux-3.x.zip](#) (sig)
- Linux (penginstal RPM) – [aws-xray-daemon-3.x.rpm](#)
- Linux (pemasang DEB) – [aws-xray-daemon-3.x.deb](#)
- Linux (ARM64, dapat dieksekusi) – [aws-xray-daemon-linux-arm64-3.x.zip](#)(sig)
- Linux (ARM64, penginstal RPM) – [aws-xray-daemon-arm64-3.x.rpm](#)
- Linux (ARM64, penginstal DEB) – [aws-xray-daemon-arm64-3.x.deb](#)
- OS X (dapat dieksekusi) – [aws-xray-daemon-macos-3.x.zip](#) (sig)
- Windows (dapat dieksekusi) – [aws-xray-daemon-windows-process-3.x.zip](#) (sig)
- Windows (layanan) – [aws-xray-daemon-windows-service-3.x.zip](#) (sig)

Tautan ini selalu mengarah ke rilis 3.x terbaru daemon. Untuk mengunduh rilis tertentu, lakukan hal berikut:

- Jika Anda ingin mengunduh rilis sebelum versi 3.3.0, ganti 3.x dengan nomor versi. Misalnya, 2.1.0. Sebelum versi 3.3.0, satu-satunya arsitektur yang tersedia adalah arm64. Misalnya, 2.1.0 dan arm64.
- Jika Anda ingin mengunduh rilis demi versi 3.3.0, ganti 3.x dengan nomor versi dan arch dengan jenis arsitektur.

Aset X-Ray direplikasi ke bucket di setiap wilayah yang didukung. Untuk menggunakan bucket yang terdekat dengan Anda atau AWS sumber daya Anda, ganti wilayah di tautan di atas dengan wilayah Anda.

```
https://s3.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-daemon/aws-xray-daemon-3.x.rpm
```

Amazon ECR

Pada versi 3.2.0 daemon dapat ditemukan di [Amazon ECR](#). Sebelum menarik gambar, Anda harus [mengautentikasi klien docker Anda](#) untuk registri publik Amazon ECR.

Tarik tanda versi 3.x terbaru yang dirilis dengan menjalankan perintah berikut:

```
docker pull public.ecr.aws/xray/aws-xray-daemon:3.x
```

Rilis sebelumnya atau alfa dapat diunduh dengan mengganti 3.x dengan alpha atau nomor versi tertentu. Tidak disarankan untuk menggunakan citra daemon tanda tag alfa di lingkungan produksi.

Docker Hub

Daemon dapat ditemukan di [Hub Docker](#). Untuk mengunduh versi 3.x terbaru yang dirilis, jalankan perintah berikut:

```
docker pull amazon/aws-xray-daemon:3.x
```

Rilis daemon sebelumnya dapat dirilis dengan mengganti 3.x dengan versi yang diinginkan.

Memverifikasi tanda tangan arsip daemon

File tanda tangan GPG disertakan untuk aset daemon yang dikompresi dalam arsip ZIP. Kunci publik ada di sini: [aws-xray.gpg](#).

Anda dapat menggunakan kunci publik untuk memverifikasi bahwa arsip ZIP daemon adalah asli dan tidak dimodifikasi. Pertama, impor kunci publik dengan [GnuPG](#).

Untuk mengimpor kunci publik

1. Unduh kunci publik.

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
$ wget $BUCKETURL/xray-daemon/aws-xray.gpg
```

2. Impor kunci publik ke dalam keyring Anda.

```
$ 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
```

Gunakan kunci yang diimpor untuk memverifikasi tanda tangan arsip ZIP daemon.

Untuk memverifikasi tanda tangan arsip

1. Unduh arsip dan file standar.

```
$ 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. Jalankan `gpg --verify` untuk memverifikasi tanda tangan.

```
$ 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
```

Catat peringatan tentang kepercayaan. Kunci hanya dapat dipercaya jika Anda atau seseorang yang Anda percaya telah menandatangani. Ini tidak berarti bahwa tanda tangan tidak valid, hanya saja Anda belum memverifikasi kunci publik.

Menjalankan daemon

Jalankan daemon secara lokal dari baris perintah. Gunakan `-o` pilihan untuk menjalankan dalam mode lokal, dan `-n` untuk mengatur wilayah.

```
~/Downloads$ ./xray -o -n us-east-2
```

Untuk petunjuk detail khusus platform, lihat topik berikut:

- Linux (lokal) – [Menjalankan daemon X-Ray di Linux](#)
- Windows (lokal) – [Menjalankan Daemon X-Ray di Windows](#)
- Elastic Beanstalk – [Menjalankan daemon X-Ray AWS Elastic Beanstalk](#)
- Amazon EC2 — [Menjalankan daemon X-Ray di Amazon EC2](#)
- Amazon ECS – [Menjalankan daemon X-Ray di Amazon ECS](#)

Anda dapat menyesuaikan perilaku daemon lebih lanjut dengan menggunakan opsi baris perintah atau file konfigurasi. Lihat [Mengkonfigurasi daemon AWS X-Ray](#) untuk detail.

Memberikan izin kepada daemon untuk mengirim data ke X-Ray

Daemon X-Ray menggunakan AWS SDK untuk mengunggah data jejak ke X-Ray, dan memerlukan AWS kredensial dengan izin untuk melakukannya.

Di Amazon EC2, daemon menggunakan peran profil instance instance secara otomatis. [Untuk informasi tentang kredensial yang diperlukan untuk menjalankan daemon secara lokal, lihat menjalankan aplikasi Anda secara lokal.](#)

Jika Anda menentukan kredensial di lebih dari satu lokasi (file kredensial, profil instans, atau variabel lingkungan), rantai penyedia SDK menentukan kredensial mana yang digunakan. Untuk informasi selengkapnya tentang memberikan kredensial ke SDK, lihat [Menentukan Kredensial](#) di AWS Panduan Pengembang SDK for Go.

IAM role atau pengguna IAM yang memiliki kredensial daemon harus memiliki izin untuk menulis data ke layanan atas nama Anda.

- Untuk menggunakan daemon di Amazon EC2, buat peran profil instance baru atau tambahkan kebijakan terkelola ke yang sudah ada.
- Untuk menggunakan daemon di Elastic Beanstalk, tambahkan kebijakan terkelola ke peran profil instans default Elastic Beanstalk.
- Untuk menjalankan daemon secara lokal, lihat [menjalankan aplikasi Anda](#) secara lokal.

Untuk informasi selengkapnya, lihat [Identitas dan manajemen akses untuk AWS X-Ray](#).

Log daemon X-Ray

Daemon mengeluarkan informasi tentang konfigurasi dan segmen saat ini yang dikirimkannya. 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)
```

Secara default, daemon mengeluarkan log ke STDOUT. Jika Anda menjalankan daemon di latar belakang, gunakan `--log-file` opsi baris perintah atau file konfigurasi untuk mengatur jalur berkas log. Anda juga dapat mengatur level log dan menonaktifkan rotasi log. Lihat [Mengkonfigurasi daemon AWS X-Ray](#) untuk instruksi.

Di Elastic Beanstalk, platform menetapkan lokasi daemon log. Lihat [Menjalankan daemon X-Ray AWS Elastic Beanstalk](#) untuk rincian selengkapnya.

Mengkonfigurasi daemon AWS X-Ray

Anda dapat menggunakan opsi baris perintah atau file konfigurasi untuk menyesuaikan perilaku daemon X-Ray. Sebagian besar pilihan tersedia menggunakan kedua metode, tetapi beberapa hanya tersedia dalam file konfigurasi dan beberapa hanya pada baris perintah.

Untuk memulai, satu-satunya pilihan yang perlu Anda ketahui adalah `-n` atau `--region`, yang Anda gunakan untuk set wilayah yang digunakan daemon untuk mengirim pelacakan data ke X-Ray.

```
~/xray-daemon$ ./xray -n us-east-2
```

Jika Anda menjalankan daemon secara lokal, yaitu, bukan di Amazon EC2, Anda dapat menambahkan `-o` opsi untuk melewati pemeriksaan kredensi profil misalnya sehingga daemon akan siap lebih cepat.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Pilihan baris perintah lainnya memungkinkan Anda mengonfigurasi pencatatan, mendengarkan pada port yang berbeda, membatasi jumlah memori yang dapat digunakan daemon, atau mengambil peran untuk mengirim pelacakan data ke akun yang berbeda.

Anda dapat melewati file konfigurasi ke daemon untuk mengakses opsi konfigurasi lanjutan dan melakukan hal-hal seperti membatasi jumlah panggilan bersamaan ke X-Ray, menonaktifkan rotasi log, dan mengirim lalu lintas ke proksi.

Bagian-bagian

- [Variabel lingkungan yang didukung](#)
- [Menggunakan opsi baris perintah](#)
- [Menggunakan file konfigurasi](#)

Variabel lingkungan yang didukung

Daemon X-Ray mendukung variabel lingkungan berikut:

- **AWS_REGION**— Menentukan titik [Wilayah AWS](#) akhir layanan X-Ray.
- **HTTPS_PROXY** – Tentukan alamat proksi untuk daemon untuk mengunggah segmen melalui. Ini bisa berupa nama domain DNS atau alamat IP dan nomor port yang digunakan oleh server proksi Anda.

Menggunakan opsi baris perintah

Lewati opsi ini ke daemon ketika Anda menjalankannya secara lokal atau dengan skrip data pengguna.

Opsi Baris Perintah

- **-b, --bind** – Dengarkan dokumen segmen pada port UDP yang berbeda.

```
--bind "127.0.0.1:3000"
```

Default —2000.

- **-t, --bind-tcp** – Dengarkan panggilan ke layanan X-Ray pada port TCP yang berbeda.

```
-bind-tcp "127.0.0.1:3000"
```

Default —2000.

- **-c, --config** – Muat file konfigurasi dari jalur yang ditentukan.

```
--config "/home/ec2-user/xray-daemon.yaml"
```

- **-f, --log-file** – Output log ke path file yang ditentukan.

```
--log-file "/var/log/xray-daemon.log"
```

- **-l, --log-level** – Tingkat log, dari yang paling bertele-tele ke paling tidak: dev, debug, info, peringatan, kesalahan, prod.

```
--log-level warn
```

Default - prod

- `-m, --buffer-memory` – Mengubah jumlah memori dalam MB yang buffer dapat menggunakan (minimal 3).

```
--buffer-memory 50
```

Default – 1% dari memori yang tersedia.

- `-o, --local-mode` — Jangan periksa EC2 metadata misalnya.
- `-r, --role-arn` – Asumsikan IAM role yang ditentukan untuk mengunggah segmen ke akun yang berbeda.

```
--role-arn "arn:aws:iam::123456789012:role/xray-cross-account"
```

- `-a, --resource-arn` — Nama Sumber Daya Amazon (ARN) dari AWS sumber daya yang menjalankan daemon.
- `-p, --proxy-address` — Unggah segmen ke AWS X-Ray melalui proxy. Protokol server proksi harus ditentukan.

```
--proxy-address "http://192.0.2.0:3000"
```

- `-n, --region` – Kirim segmen ke layanan X-Ray di wilayah tertentu.
- `-v, --version` — Tampilkan versi AWS X-Ray daemon.
- `-h, --help` – Tampilkan layar bantuan.

Menggunakan file konfigurasi

Anda juga dapat menggunakan file format YAML untuk mengonfigurasi daemon. Lewati file konfigurasi ke daemon dengan menggunakan `-c` Pilihan.

```
~$ ./xray -c ~/xray-daemon.yaml
```

Opsi file konfigurasi

- `TotalBufferSizeMB` – Ukuran buffer maksimum dalam MB (minimal 3). Pilih 0 untuk menggunakan 1% dari memori host.

- **Concurrency**— Jumlah maksimum panggilan bersamaan untuk AWS X-Ray mengunggah dokumen segmen.
- **Region**— Kirim segmen ke AWS X-Ray layanan di wilayah tertentu.
- **Socket** – Konfigurasi pengikatan daemon.
 - **UDPEndPoint** – Ubah port tempat daemon mendengarkan.
 - **TCPAddress** – Dengarkan [panggilan ke layanan X-Ray](#) pada port TCP yang berbeda.
- **Logging** – Konfigurasi perilaku pencatatan.
 - **LogRotation** – Set ke `false` untuk menonaktifkan rotasi log.
 - **LogLevel**— Ubah level log, dari yang paling bertele-tele menjadi paling sedikit: `dev`, `debug`, `info` atau `prod`, `warn`, `error`. Secara default adalah `prod`, yang setara dengan `info`.
 - **LogPath** – Output log ke jalur file yang ditentukan.
- **LocalMode**— Atur `true` untuk melewati pemeriksaan EC2 misalnya metadata.
- **ResourceARN**— Nama Sumber Daya Amazon (ARN) dari AWS sumber daya yang menjalankan daemon.
- **RoleARN** – Asumsikan IAM role yang ditentukan untuk mengunggah segmen ke akun yang berbeda.
- **ProxyAddress**— Unggah segmen ke AWS X-Ray melalui proxy.
- **Endpoint** – Ubah titik akhir layanan X-Ray untuk tempat daemon mengirimkan dokumen segmen.
- **NoVerifySSL** – Nonaktifkan verifikasi sertifikat TLS.
- **Version** – Versi format file konfigurasi daemon. Versi format file adalah wajib bidang.

Example Xray-daemon.yaml

File konfigurasi ini mengubah port mendengarkan daemon menjadi 3000, mematikan pemeriksaan metadata instans, set peran yang akan digunakan untuk mengunggah segmen, dan mengubah wilayah dan opsi pencatatan.

```
Socket:
  UDPEndPoint: "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
```

Menjalankan daemon X-Ray secara lokal

Anda dapat menjalankan AWS X-Ray daemon secara lokal di Linux, macOS, Windows, atau dalam wadah Docker. Jalankan daemon untuk menyampaikan data pelacakan ke X-Ray saat Anda mengembangkan dan menguji aplikasi Anda yang diinstrumentasi. Unduh dan ekstrak daemon dengan menggunakan petunjuk [di sini](#).

Saat berjalan secara lokal, daemon dapat membaca kredensial dari file kredensial AWS SDK (.aws/credentials di direktori pengguna Anda) atau dari variabel lingkungan. Untuk informasi selengkapnya, lihat [Memberikan izin kepada daemon untuk mengirim data ke X-Ray](#).

Daemon mendengarkan data UDP pada port 2000. Anda dapat mengubah port dan opsi lain dengan menggunakan file konfigurasi dan opsi baris perintah. Untuk informasi selengkapnya, lihat [Mengkonfigurasi daemon AWS X-Ray](#).

Menjalankan daemon X-Ray di Linux

Anda dapat menjalankan daemon yang dapat dieksekusi dari baris perintah. Gunakan opsi `-o` untuk menjalankan dalam mode lokal, dan `-n` untuk mengatur wilayah.

```
~/xray-daemon$ ./xray -o -n us-east-2
```

Untuk menjalankan daemon di latar belakang, gunakan `&`.

```
~/xray-daemon$ ./xray -o -n us-east-2 &
```

Hentikan proses daemon yang berjalan di latar belakang dengan `pkill`.

```
~$ pkill xray
```

Menjalankan daemon X-Ray di kontainer Docker

Untuk menjalankan daemon secara lokal di kontainer Docker, simpan teks berikut ke file bernama `Dockerfile`. Unduh [contoh gambar](#) lengkap di Amazon ECR. Lihat [mengunduh daemon untuk informasi](#) lebih lanjut.

Example Dockerfile – Amazon Linux

```
FROM amazonlinux
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

Bangun citra kontainer dengan `docker build`.

```
~/xray-daemon$ docker build -t xray-daemon .
```

Jalankan citra dalam kontainer dengan `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
```

Perintah ini menggunakan opsi berikut:

- `--attach STDOUT` – Lihat output dari daemon di terminal.
- `-v ~/.aws/:/root/.aws/:ro`— Berikan akses read-only container ke `.aws` direktori untuk membiarkannya membaca kredensial AWS SDK Anda.
- `AWS_REGION=us-east-2` – Set variabel lingkungan `AWS_REGION` untuk memberitahu daemon wilayah yang akan digunakan.
- `--net=host` – Lampirkan kontainer ke jaringan host. Kontainer pada jaringan host dapat berkomunikasi satu sama lain tanpa memublikasikan port.
- `-p 2000:2000/udp` – Petakan UDP port 2000 pada mesin Anda ke port yang sama pada kontainer. Ini tidak diperlukan untuk kontainer di jaringan yang sama untuk berkomunikasi, tetapi memungkinkan Anda mengirim segmen ke daemon [dari baris perintah](#) atau dari aplikasi yang tidak berjalan di Docker.

- `--name xray-daemon` – Namakan kontainer `xray-daemon` bukan menghasilkan nama acak.
- `-o` (setelah nama citra) - Tambahkan opsi `-o` ke titik masuk yang menjalankan daemon dalam kontainer. Opsi ini memberi tahu daemon untuk berjalan dalam mode lokal untuk mencegahnya mencoba membaca metadata EC2 instance Amazon.

Untuk menghentikan daemon, gunakan `docker stop`. Jika Anda membuat perubahan pada `Dockerfile` dan membangun citra baru, Anda perlu menghapus kontainer yang sudah ada sebelum Anda dapat membuat kontainer lain dengan nama yang sama. Gunakan `docker rm` untuk menghapus kontainer.

```
$ docker stop xray-daemon
$ docker rm xray-daemon
```

Menjalankan Daemon X-Ray di Windows

Anda dapat menjalankan daemon yang dapat dieksekusi dari baris perintah. Gunakan opsi `-o` untuk menjalankan dalam mode lokal, dan `-n` untuk mengatur wilayah.

```
> .\xray_windows.exe -o -n us-east-2
```

Gunakan PowerShell skrip untuk membuat dan menjalankan layanan untuk daemon.

Example PowerShell skrip - 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
```

Menjalankan daemon X-Ray di OS X

Anda dapat menjalankan daemon yang dapat dieksekusi dari baris perintah. Gunakan opsi `-o` untuk menjalankan dalam mode lokal, dan `-n` untuk mengatur wilayah.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2
```

Untuk menjalankan daemon di latar belakang, gunakan `&`.

```
~/xray-daemon$ ./xray_mac -o -n us-east-2 &
```

Gunakan `nohup` untuk mencegah daemon berhenti ketika terminal ditutup.

```
~/xray-daemon$ nohup ./xray_mac &
```

Menjalankan daemon X-Ray AWS Elastic Beanstalk

Untuk menyampaikan data pelacakan dari aplikasi Anda ke AWS X-Ray, Anda dapat menjalankan daemon X-Ray pada instans Amazon lingkungan Elastic Beanstalk Anda. EC2 Untuk daftar platform yang didukung, lihat [Mengonfigurasi AWS X-Ray Debugging](#) di AWS Elastic Beanstalk Panduan Developer.

Note

Daemon tersebut menggunakan profil instans lingkungan Anda untuk izin. Untuk petunjuk tentang penambahan izin ke profil instans Elastic Beanstalk, lihat [Memberikan izin kepada daemon untuk mengirim data ke X-Ray](#).

Platform Elastic Beanstalk menyediakan opsi konfigurasi yang dapat Anda atur untuk menjalankan daemon secara otomatis. Anda dapat mengaktifkan daemon dalam file konfigurasi dalam kode

sumber Anda atau dengan memilih opsi di konsol Elastic Beanstalk. Bila Anda mengaktifkan opsi konfigurasi, daemon diinstal pada instans dan berjalan sebagai layanan.

Versi yang disertakan pada platform Elastic Beanstalk mungkin bukan versi terbaru. Lihat [Platform yang didukung topik](#) untuk mengetahui versi daemon yang tersedia untuk konfigurasi platform Anda.

Elastic Beanstalk tidak menyediakan X-Ray daemon pada platform Multicontainer Docker (Amazon ECS).

Menggunakan integrasi X-Ray Elastic Beanstalk untuk menjalankan X-Ray daemon

Gunakan konsol tersebut untuk mengaktifkan integrasi X-Ray, atau konfigurasikan di kode sumber aplikasi Anda dengan file konfigurasi.

Untuk mengaktifkan X-Ray daemon di konsol Elastic Beanstalk

1. Buka [konsol Elastic Beanstalk](#).
2. Arahkan ke [konsol manajemen](#) untuk lingkungan Anda.
3. Pilih Konfigurasi.
4. Pilih Pengaturan Perangkat Lunak.
5. Untuk X-Ray daemon, pilih Diaktifkan.
6. Pilih Terapkan.

Anda dapat menyertakan file konfigurasi dalam kode sumber Anda untuk membuat konfigurasi portabel antar lingkungan.

Example `.ebextensions/xray-daemon.config`

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

Elastic Beanstalk melewati file konfigurasi ke daemon dan output log ke lokasi standar.

Pada Platform Server Windows

- file konfigurasi – `C:\Program Files\Amazon\XRay\cfg.yaml`

- Log – c:\Program Files\Amazon\XRay\logs\xray-service.log

Pada Platform Linux

- File konfigurasi – /etc/amazon/xray/cfg.yaml
- Log – /var/log/xray/xray.log

Elastic Beanstalk menyediakan alat untuk menarik log AWS Management Console instance dari atau baris perintah. Anda dapat memberitahu Elastic Beanstalk untuk memasukkan log X-Ray daemon dengan menambahkan tugas dengan file konfigurasi.

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 - server Windows

```
files:
  "c:/Program Files/Amazon/ElasticBeanstalk/config/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      c:\Program Files\Amazon\XRay\logs\xray-service.log
```

Lihat [Melihat Log dari AWS Elastic Beanstalk Instans Amazon Lingkungan Elastis Beanstalk Anda di Panduan EC2 Pengembang](#) untuk informasi selengkapnya.

Mengunduh dan menjalankan daemon X-Ray secara manual (lanjutan)

Jika daemon X-Ray tidak tersedia untuk konfigurasi platform Anda, Anda dapat mengunduhnya dari Amazon S3 dan jalankan dengan file konfigurasi.

Gunakan file konfigurasi Elastic Beanstalk untuk mengunduh dan menjalankan daemon.

Example .ebextensions/xray.config - Linux

```

commands:
  01-stop-tracing:
    command: yum remove -y xray
    ignoreErrors: true
  02-copy-tracing:
    command: curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
  03-start-tracing:
    command: yum install -y /home/ec2-user/xray.rpm

files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      /var/log/xray/xray.log
  "/etc/amazon/xray/cfg.yaml" :
    mode: "000644"
    owner: root
    group: root
    content: |
      Logging:
        LogLevel: "debug"
        Version: 2

```

Example .ebextensions/xray.config - server Windows

```

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

```

Contoh-contoh ini juga menambahkan berkas log daemon ke tugas log ekor Elastic Beanstalk, sehingga disertakan ketika Anda meminta log dengan konsol tersebut atau Elastic Beanstalk Command Line Interface (EB CLI).

Menjalankan daemon X-Ray di Amazon EC2

Anda dapat menjalankan daemon X-Ray pada sistem operasi berikut di Amazon: EC2

- Amazon Linux
- Ubuntu

- Server Windows (2012 R2 dan yang lebih baru)

Gunakan profil instans untuk memberikan daemon izin untuk mengunggah data pelacakan ke X-Ray. Untuk informasi selengkapnya, lihat [Memberikan izin kepada daemon untuk mengirim data ke X-Ray](#).

Gunakan skrip data pengguna untuk menjalankan daemon secara otomatis saat Anda meluncurkan instans.

Example Skrip data pengguna - 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 Skrip data pengguna - Server Windows

```
<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>
```

Menjalankan daemon X-Ray di Amazon ECS

Di Amazon ECS, buat citra Docker yang menjalankan daemon X-Ray, unggah ke repositori citra Docker, lalu deploy ke kluster Amazon ECS Anda. Anda dapat menggunakan pemetaan port dan pengaturan mode jaringan dalam file ketentuan tugas Anda untuk memungkinkan aplikasi Anda berkomunikasi dengan kontainer daemon.

Menggunakan citra Docker resmi

X-Ray menyediakan [image kontainer](#) Docker di Amazon ECR yang dapat Anda terapkan bersama aplikasi Anda. Lihat [mengunduh daemon untuk informasi](#) lebih lanjut.

Example Ketentuan tugas

```
{
  "name": "xray-daemon",
  "image": "amazon/aws-xray-daemon",
  "cpu": 32,
  "memoryReservation": 256,
  "portMappings" : [
    {
      "hostPort": 0,
      "containerPort": 2000,
      "protocol": "udp"
    }
  ]
}
```

Buat dan bangun citra Docker

Untuk konfigurasi kustom, Anda mungkin perlu menentukan citra Docker milik Anda sendiri.

Tambahkan kebijakan terkelola ke peran tugas Anda untuk memberikan izin kepada daemon untuk mengunggah data pelacakan ke X-Ray. Untuk informasi selengkapnya, lihat [Memberikan izin kepada daemon untuk mengirim data ke X-Ray](#).

Gunakan salah satu file Docker berikut untuk membuat citra yang menjalankan daemon.

Example Dockerfile – Amazon Linux

```
FROM amazonlinux
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

Note

Flag `-t` dan `-b` diperlukan untuk menentukan alamat pengikatan untuk mendengarkan loopback dari lingkungan multi-kontainer.

Example Dockerfile – Ubuntu

Untuk turunan Debian, Anda juga perlu menginstal sertifikat otoritas sertifikasi (CA) untuk menghindari masalah saat mengunduh penginstal.

```
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
```

Dalam ketentuan tugas Anda, konfigurasi tergantung pada mode jaringan yang Anda gunakan. Jaringan jembatan adalah default dan dapat digunakan di VPC default Anda. Di jaringan jembatan,

atur variabel lingkungan `AWS_XRAY_DAEMON_ADDRESS` untuk memberi tahu SDK X-Ray, port kontainer yang akan dirujuk dan mengatur port host. Misalnya, Anda dapat memublikasikan UDP port 2000, dan membuat tautan dari kontainer aplikasi Anda ke kontainer daemon.

Example Ketentuan tugas

```
{
  "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"
  ]
}
```

Jika Anda menjalankan kluster di subnet privat VPC, Anda dapat menggunakan [mode jaringan awsvpc](#) untuk melampirkan antarmuka jaringan elastis (ENI) ke kontainer Anda. Hal ini memungkinkan Anda untuk menghindari penggunaan tautan. Abaikan port host di pemetaan port, tautan, dan variabel lingkungan AWS_XRAY_DAEMON_ADDRESS.

Example Ketentuan tugas 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
        }
      ]
    }
  ]
}
```

Konfigurasi opsi baris perintah di konsol Amazon ECS

Opsi baris perintah mengganti nilai-nilai yang bertentangan dalam file konfigurasi citra Anda.

Opsi baris perintah biasanya digunakan untuk pengujian lokal, tetapi juga dapat digunakan untuk memudahkan saat mengatur variabel lingkungan, atau mengontrol proses startup.

Dengan menambahkan opsi baris perintah, Anda memperbarui CMD Docker yang diteruskan ke kontainer. Untuk informasi selengkapnya, lihat [Referensi menjalankan Docker](#).

Untuk mengatur opsi baris perintah

1. Buka konsol klasik Amazon ECS di <https://console.aws.amazon.com/ecs/>.
2. Dari bilah navigasi, pilih wilayah yang berisi ketentuan tugas Anda.
3. Di panel navigasi, pilih Ketentuan Tugas.
4. Di halaman Ketentuan Tugas, pilih kotak di sebelah kiri ketentuan tugas yang akan direvisi dan pilih Buat revisi baru.
5. Di halaman Buat revisi baru dari Ketentuan Tugas, pilih kontainer.
6. Di bagian LINGKUNGAN, tambahkan daftar Anda yang dipisahkan koma dari opsi baris perintah ke bidang Perintah.
7. Pilih Perbarui.
8. Verifikasi informasi dan pilih Buat.

Contoh berikut menunjukkan cara menulis opsi baris perintah yang dipisahkan koma untuk opsi RoleARN. Opsi RoleARN mengasumsikan IAM role yang ditentukan untuk mengunggah segmen ke akun yang berbeda.

Example

```
--role-arn, arn:aws:iam::123456789012:role/xray-cross-account
```

Untuk mempelajari lebih lanjut tentang opsi baris perintah yang tersedia di X-Ray, lihat [Mengonfigurasi AWS X-Ray Daemon](#).

Integrasi AWS X-Ray dengan yang lain Layanan AWS

Banyak yang Layanan AWS menyediakan berbagai tingkat integrasi X-Ray, termasuk pengambilan sampel dan penambahan header ke permintaan yang masuk, menjalankan daemon X-Ray, dan secara otomatis mengirim data jejak ke X-Ray. Integrasi dengan X-Ray dapat mencakup hal-hal berikut:

- Instrumentasi aktif — Sampel dan instrumen permintaan masuk
- Instrumentasi pasif — Permintaan instrumen yang telah diambil sampelnya oleh layanan lain
- Request tracing - Menambahkan header tracing ke semua permintaan yang masuk dan menyebarkannya ke hilir
- Tooling — Menjalankan daemon X-Ray untuk menerima segmen dari X-Ray SDK

Note

X-Ray SDKs menyertakan plugin untuk integrasi tambahan dengan Layanan AWS. Misalnya, Anda dapat menggunakan plugin X-Ray SDK for Java Elastic Beanstalk untuk menambahkan informasi tentang lingkungan Elastic Beanstalk yang menjalankan aplikasi Anda, termasuk nama lingkungan dan ID.

Berikut adalah beberapa contoh Layanan AWS yang terintegrasi dengan X-Ray:

- [AWS Distro for OpenTelemetry \(ADOT\)](#) — Dengan ADOT, para insinyur dapat menginstruksikan aplikasi mereka sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan termasuk Amazon, Amazon Service CloudWatch OpenSearch , dan AWS X-Ray Amazon Managed Service untuk Prometheus.
- [AWS Lambda](#)— Instrumentasi aktif dan pasif dari permintaan masuk pada semua runtime. AWS Lambda menambahkan dua node ke peta jejak Anda, satu untuk AWS Lambda layanan, dan satu untuk fungsi. Saat Anda mengaktifkan instrumentasi, jalankan AWS Lambda juga daemon X-Ray di runtime Java dan Node.js untuk digunakan dengan X-Ray SDK.
- [Amazon API Gateway](#) – Instrumentasi aktif dan pasif. API Gateway menggunakan aturan pengambilan sampel untuk menentukan permintaan yang dicatat, dan menambahkan simpul untuk tahap gateway ke peta layanan Anda.
- [AWS Elastic Beanstalk](#) - Alat. Elastic Beanstalk memiliki daemon X-Ray pada platform berikut:

- Java SE – 2.3.0 dan konfigurasi yang lebih baru
- Tomcat – 2.4.0 dan konfigurasi yang lebih baru
- Node.js – 3.2.0 dan konfigurasi yang lebih baru
- Windows Server — Semua konfigurasi selain Windows Server Core yang telah dirilis setelah 9 Desember 2016

Anda dapat menggunakan konsol Elastic Beanstalk untuk memberitahu Elastic Beanstalk agar menjalankan daemon pada platform ini, atau menggunakan opsi `XRayEnabled` di namespace `aws:elasticbeanstalk:xray`.

- [Elastic Load Balancing](#) — Permintaan penelusuran pada Application Load Balancers. Application Load Balancer menambahkan ID jejak ke header permintaan sebelum mengirimnya ke grup target.
- [Amazon EventBridge](#) - Instrumentasi pasif. Jika layanan yang menerbitkan peristiwa ke diinstrumentasi dengan EventBridge X-Ray SDK, target peristiwa akan menerima header penelusuran dan dapat terus menyebarkan ID jejak asli.
- [Amazon Simple Notification Service](#) – Instrumentasi pasif. Jika penerbit Amazon SNS melacak kliennya dengan X-Ray SDK, pelanggan dapat mengambil header pelacakan dan terus menyebarkan pelacakan asli dari penerbit dengan ID pelacakan yang sama.
- [Amazon Simple Queue Service](#) – Instrumentasi pasif. Jika suatu layanan melacak permintaan menggunakan X-Ray SDK, Amazon SQS dapat mengirim header pelacakan dan terus menyebarkan pelacakan asli dari pengirim ke konsumen dengan ID pelacakan yang konsisten.

Pilih dari topik berikut untuk menjelajahi set lengkap terintegrasi Layanan AWS.

Topik

- [AWS Distro untuk OpenTelemetry dan AWS X-Ray](#)
- [Dukungan penelusuran aktif Amazon API Gateway untuk AWS X-Ray](#)
- [Amazon EC2 dan AWS App Mesh](#)
- [AWS Pelari Aplikasi dan X-Ray](#)
- [AWS AppSync dan AWS X-Ray](#)
- [Mencatat panggilan X-Ray API dengan AWS CloudTrail](#)
- [CloudWatch Integrasi dengan X-Ray](#)
- [Melacak perubahan konfigurasi enkripsi X-Ray AWS Config](#)
- [Amazon Elastic Compute Cloud dan AWS X-Ray](#)

- [AWS Elastic Beanstalk dan AWS X-Ray](#)
- [Elastic Load Balancing dan AWS X-Ray](#)
- [Amazon EventBridge dan AWS X-Ray](#)
- [AWS Lambda dan AWS X-Ray](#)
- [Amazon SNS dan AWS X-Ray](#)
- [AWS Step Functions dan AWS X-Ray](#)
- [Amazon SQS dan AWS X-Ray](#)
- [Amazon S3 dan AWS X-Ray](#)

AWS Distro untuk OpenTelemetry dan AWS X-Ray

Gunakan AWS Distro for OpenTelemetry (ADOT) untuk mengumpulkan dan mengirim metrik dan jejak ke AWS X-Ray dan solusi pemantauan lainnya, seperti Amazon, Layanan Amazon, dan OpenSearch Layanan Terkelola CloudWatch Amazon untuk Prometheus.

AWS Distro untuk OpenTelemetry

AWS Distro for OpenTelemetry (ADOT) adalah AWS distribusi berdasarkan proyek Cloud Native Computing Foundation (CNCF). OpenTelemetry OpenTelemetry menyediakan satu set open source APIs, perpustakaan, dan agen untuk mengumpulkan jejak dan metrik terdistribusi. Toolkit ini adalah distribusi OpenTelemetry komponen hulu termasuk SDKs, agen instrumentasi otomatis, dan kolektor yang diuji, dioptimalkan, diamankan, dan didukung oleh. AWS

Dengan ADOT, para insinyur dapat menginstruksikan aplikasi mereka sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa solusi AWS pemantauan termasuk Amazon CloudWatch, Amazon Service, dan AWS X-Ray Amazon Managed OpenSearch Service untuk Prometheus.

ADOT terintegrasi dengan semakin banyak Layanan AWS untuk menyederhanakan pengiriman jejak dan metrik ke solusi pemantauan seperti X-Ray. Beberapa contoh layanan yang terintegrasi dengan ADOT meliputi:

- **AWS Lambda**— Lapisan Lambda yang AWS dikelola untuk ADOT memberikan pengalaman plug-and-play pengguna dengan secara otomatis menginstrumentasi fungsi Lambda, OpenTelemetry mengemas bersama dengan out-of-the-box konfigurasi untuk AWS Lambda dan X-Ray dalam lapisan yang mudah diatur. Pengguna dapat mengaktifkan dan menonaktifkan OpenTelemetry fungsi Lambda mereka tanpa mengubah kode. Untuk informasi lebih lanjut, lihat [AWS Distro untuk OpenTelemetry](#) Lambda

- Amazon Elastic Container Service (ECS) — Kumpulkan metrik dan jejak dari aplikasi Amazon ECS menggunakan AWS Distro for OpenTelemetry Collector, untuk dikirim ke X-Ray dan solusi pemantauan lainnya. Untuk informasi selengkapnya, lihat [Mengumpulkan data jejak aplikasi](#) di panduan pengembang Amazon ECS.
- AWS App Runner — App Runner mendukung pengiriman jejak ke X-Ray menggunakan AWS Distro for OpenTelemetry (ADOT). Gunakan ADOT SDKs untuk mengumpulkan data jejak untuk aplikasi kontainer Anda, dan gunakan X-Ray untuk menganalisis dan mendapatkan wawasan tentang aplikasi instrumentasi Anda. Untuk informasi selengkapnya, lihat [Pelari AWS Aplikasi dan X-Ray](#).

Untuk informasi selengkapnya tentang AWS Distro for OpenTelemetry, termasuk integrasi dengan tambahan Layanan AWS, lihat [AWS Distro untuk OpenTelemetry](#) Dokumentasi.

Untuk informasi selengkapnya tentang menginstrumentasi aplikasi Anda dengan AWS Distro for dan OpenTelemetry X-Ray, lihat [Menginstrumentasi aplikasi Anda dengan Distro untuk AWS](#).
OpenTelemetry

Dukungan penelusuran aktif Amazon API Gateway untuk AWS X-Ray

Anda dapat menggunakan X-Ray untuk melacak dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui Amazon API Gateway Anda APIs ke layanan yang mendasarinya. API Gateway mendukung penelusuran X-Ray untuk semua tipe titik akhir API Gateway: Regional, edge-optimized, dan privat. Anda dapat menggunakan X-Ray dengan Amazon API Gateway di semua Wilayah AWS tempat X-Ray tersedia. Untuk informasi selengkapnya, lihat [Menelusuri Eksekusi API dari API Gateway dengan AWS X-Ray](#) di Panduan Developer Amazon API Gateway.

Note

X-Ray hanya mendukung penelusuran REST APIs melalui API Gateway.

Amazon API Gateway menyediakan dukungan [penelusuran aktif](#) untuk AWS X-Ray. Aktifkan penelusuran aktif pada tahap API Anda untuk mengambil sampel permintaan masuk dan mengirim penelusuran ke X-Ray.

Untuk mengaktifkan penelusuran aktif pada tahap API

1. Buka konsol API Gateway di <https://console.aws.amazon.com/apigateway/>.
2. Pilih API.
3. Memilih tahapan.
4. Pada tab Logs/Tracing, pilih Aktifkan X-Ray Tracing dan kemudian pilih Simpan Perubahan.
5. Pilih Sumber Daya pada panel navigasi sebelah kiri.
6. Untuk menerapkan ulang API dengan pengaturan baru, pilih dropdown Tindakan, lalu pilih Deploy API.

API Gateway menggunakan aturan pengambilan sampel yang Anda tetapkan di konsol X-Ray untuk menentukan permintaan mana yang akan dicatat. Anda dapat membuat aturan yang hanya berlaku untuk APIs, atau yang hanya berlaku untuk permintaan yang berisi header tertentu. API Gateway mencatat header dalam atribut pada segmen, bersama dengan detail tentang tahap dan permintaan. Untuk informasi selengkapnya, lihat [Mengonfigurasi aturan pengambilan sampel](#).

Note

Saat melacak REST APIs dengan [integrasi HTTP](#) API Gateway, nama layanan setiap segmen disetel ke jalur URL permintaan dari API Gateway ke titik akhir integrasi HTTP Anda, menghasilkan node layanan pada peta jejak X-Ray untuk setiap jalur URL unik. Sejumlah besar jalur URL dapat menyebabkan peta jejak melebihi batas 10.000 node, yang mengakibatkan kesalahan.

Untuk meminimalkan jumlah node layanan yang dibuat oleh API Gateway, pertimbangkan untuk meneruskan parameter dalam string kueri URL atau di badan permintaan melalui POST. Pendekatan mana pun akan memastikan parameter bukan bagian dari jalur URL, yang dapat menghasilkan lebih sedikit jalur URL dan node layanan yang berbeda.

Untuk semua permintaan yang masuk, API Gateway menambahkan [header pelacakan](#) ke permintaan HTTP masuk yang belum memilikinya.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```


Format ID jejak X-Ray

`trace_id`X-Ray terdiri dari tiga angka yang dipisahkan oleh tanda hubung. Contohnya, `1-58406520-a006649127e371903a2de979`. Hal ini mencakup:

- Nomor versi, yaitu 1.
- Waktu permintaan asli dalam waktu epoch Unix menggunakan 8 digit heksadesimal.

Misalnya, 10:00 AM 1 Desember 2016 PST dalam waktu epoch adalah 1480615200 detik atau 58406520 dalam digit heksadesimal.

- Pengidentifikasi 96-bit yang unik secara global untuk jejak dalam 24 digit heksadesimal.

Jika penelusuran aktif dinonaktifkan, tahapan masih mencatat segmen jika permintaan berasal dari layanan yang mengambil sampel permintaan dan mulai penelusuran. Misalnya, aplikasi web yang diinstrumentasi dapat memanggil API dari API Gateway dengan klien HTTP. Ketika Anda instrumen klien HTTP dengan SDK X-Ray, hal tersebut akan menambahkan header penelusuran ke permintaan keluar yang berisi keputusan pengambilan sampel. API Gateway membaca header penelusuran dan membuat segmen untuk permintaan sampel.

Jika Anda menggunakan API Gateway untuk [membuat Java SDK untuk API Anda](#), Anda dapat menginstrumentasikan klien SDK dengan menambahkan penanganan permintaan dengan pembuat klien, dengan cara yang sama seperti Anda akan secara manual AWS menginstruksikan klien SDK. Lihat [Menelusuri panggilan AWS SDK dengan X-Ray SDK for Java](#) untuk instruksi.

Amazon EC2 dan AWS App Mesh

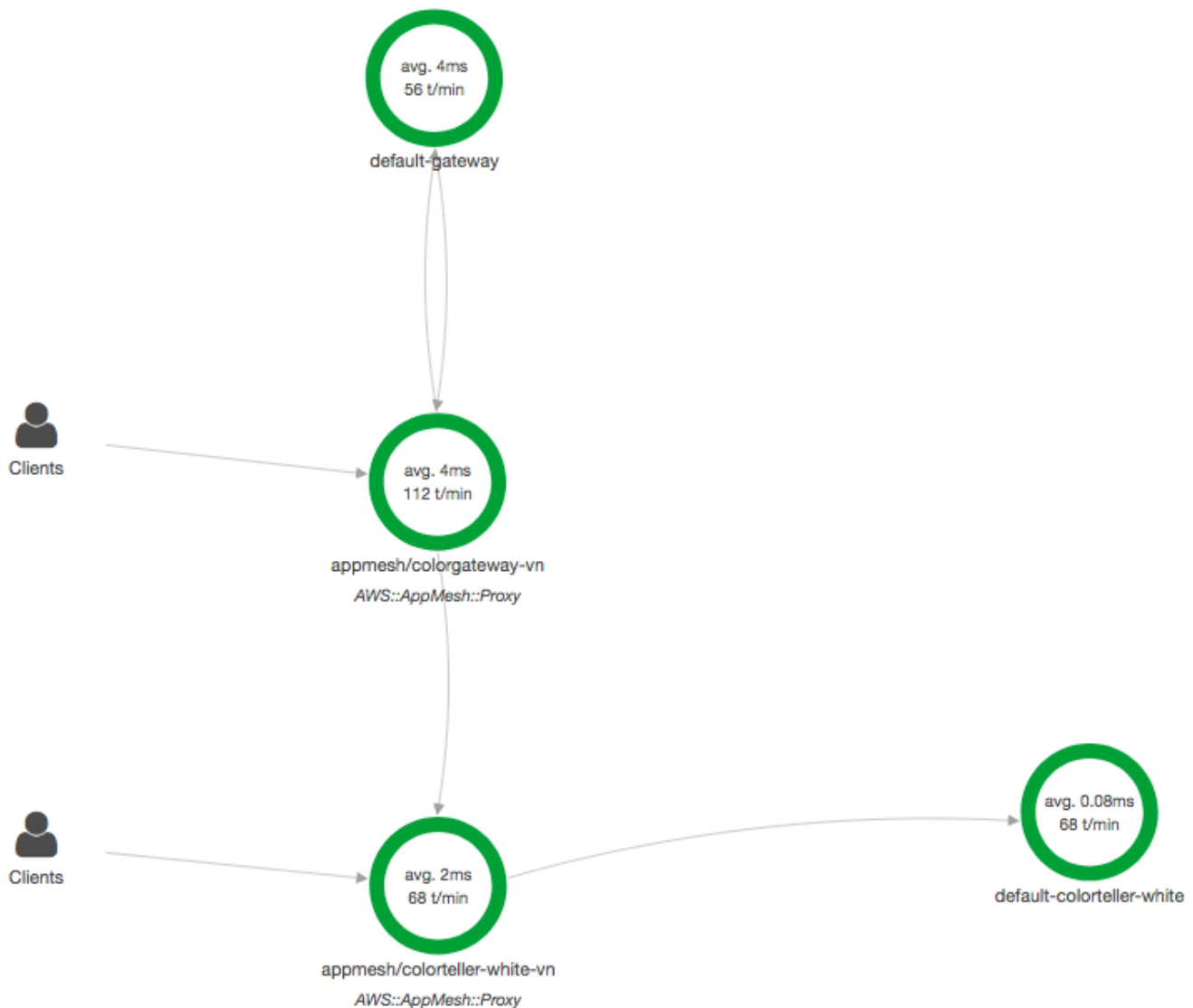
AWS X-Ray terintegrasi dengan [AWS App Mesh](#) untuk mengelola proxy Utusan untuk layanan mikro. App Mesh menyediakan versi Envoy yang dapat Anda konfigurasi untuk mengirim data penelusuran ke daemon X-Ray yang berjalan dalam kontainer tugas atau pod yang sama. X-Ray mendukung penelusuran dengan layanan yang kompatibel dengan App Mesh berikut ini:

- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Amazon Elastic Compute Cloud (Amazon EC2)

Gunakan petunjuk berikut untuk mempelajari cara mengaktifkan penelusuran X-Ray melalui App Mesh.

Service map

Enter a service name to find and select the node on map



Untuk mengonfigurasi proksi Envoy untuk mengirim data ke X-Ray, atur `ENABLE_ENVOY_XRAY_TRACING` [Variabel Lingkungan](#) dalam ketentuan kontainer nya.

Note

Versi App Mesh dari Envoy saat ini tidak mengirim jejak berdasarkan aturan [pengambilan sampel](#) yang dikonfigurasi. Sebaliknya, ia menggunakan tingkat pengambilan sampel tetap

5% untuk Utusan versi 1.16.3 atau yang lebih baru, atau tingkat pengambilan sampel 50% untuk versi Envoy sebelum 1.16.3.

Example Envoy ketentuan kontainer untuk 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

Untuk mempelajari selengkapnya tentang alamat wilayah Envoy yang tersedia, lihat [Envoy citra](#) di Panduan Pengguna AWS App Mesh .

Untuk detail tentang menjalankan daemon X-Ray dalam kontainer, lihat [Menjalankan daemon X-Ray di Amazon ECS](#). [Untuk contoh aplikasi yang menyertakan mesh layanan, microservice, proxy Envoy, dan daemon X-Ray, terapkan sampel di repositori Contoh App Mesh. colorapp GitHub](#)

Pelajari Selengkapnya

- [Memulai dengan AWS App Mesh](#)
- [Memulai dengan AWS App Mesh dan Amazon ECS](#)

AWS Pelari Aplikasi dan X-Ray

AWS App Runner adalah aplikasi Layanan AWS yang menyediakan cara yang cepat, sederhana, dan hemat biaya untuk menyebarkan dari kode sumber atau gambar kontainer langsung ke aplikasi web yang dapat diskalakan dan aman di aplikasi. AWS Cloud Anda tidak perlu mempelajari teknologi baru, memutuskan layanan komputasi mana yang akan digunakan, atau mengetahui cara menyediakan dan mengonfigurasi AWS sumber daya. Lihat [Apa itu Pelari AWS Aplikasi](#) untuk informasi selengkapnya.

AWS App Runner mengirimkan jejak ke X-Ray dengan mengintegrasikan dengan [AWS Distro for OpenTelemetry](#) (ADOT). Gunakan ADOT SDKs untuk mengumpulkan data jejak untuk aplikasi kontainer Anda, dan gunakan X-Ray untuk menganalisis dan mendapatkan wawasan tentang aplikasi instrumentasi Anda. Untuk informasi selengkapnya, lihat [Menelusuri aplikasi App Runner Anda dengan X-Ray](#).

AWS AppSync dan AWS X-Ray

Anda dapat mengaktifkan dan melacak permintaan untuk AWS AppSync. Untuk informasi lebih lanjut, lihat [Menelusuri dengan AWS X-Ray](#) untuk instruksi.

Saat penelusuran X-Ray diaktifkan untuk AWS AppSync API, [peran terkait layanan AWS](#) Identity and Access Management secara otomatis dibuat di akun Anda dengan izin yang sesuai. Ini memungkinkan AWS AppSync untuk mengirim jejak ke X-Ray dengan cara yang aman.

Mencatat panggilan X-Ray API dengan AWS CloudTrail

AWS X-Ray terintegrasi dengan [AWS CloudTrail](#), layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS. CloudTrail menangkap semua panggilan API untuk X-Ray sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari konsol X-Ray dan panggilan kode ke operasi X-Ray API. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk X-Ray, alamat IP dari mana permintaan dibuat, kapan dibuat, dan detail tambahan.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut hal ini:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau pengguna.
- Apakah permintaan dibuat atas nama pengguna IAM Identity Center.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

CloudTrail aktif di Anda Akun AWS ketika Anda membuat akun dan Anda secara otomatis memiliki akses ke riwayat CloudTrail Acara. Riwayat CloudTrail Acara menyediakan catatan yang dapat dilihat, dapat dicari, dapat diunduh, dan tidak dapat diubah dari 90 hari terakhir dari peristiwa manajemen yang direkam dalam file. Wilayah AWS Untuk informasi selengkapnya, lihat [Bekerja dengan riwayat CloudTrail Acara](#) di Panduan AWS CloudTrail Pengguna. Tidak ada CloudTrail biaya untuk melihat riwayat Acara.

Untuk catatan acara yang sedang berlangsung dalam 90 hari Akun AWS terakhir Anda, buat jejak atau penyimpanan data acara [CloudTrailDanau](#).

CloudTrail jalan setapak

Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Semua jalur yang dibuat menggunakan AWS Management Console Multi-region. Anda dapat membuat jalur Single-region atau Multi-region dengan menggunakan. AWS CLI Membuat jejak Multi-wilayah disarankan karena Anda menangkap aktivitas Wilayah AWS di semua akun Anda. Jika Anda membuat jejak wilayah Tunggal, Anda hanya dapat melihat peristiwa yang dicatat di jejak. Wilayah AWS Untuk informasi selengkapnya tentang jejak, lihat [Membuat jejak untuk Anda Akun AWS](#) dan [Membuat jejak untuk organisasi](#) di Panduan AWS CloudTrail Pengguna.

Anda dapat mengirimkan satu salinan acara manajemen yang sedang berlangsung ke bucket Amazon S3 Anda tanpa biaya CloudTrail dengan membuat jejak, namun, ada biaya penyimpanan Amazon S3. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#). Untuk informasi tentang harga Amazon S3, lihat [Harga Amazon S3](#).

CloudTrail Penyimpanan data acara danau

CloudTrail Lake memungkinkan Anda menjalankan kueri berbasis SQL pada acara Anda. CloudTrail [Lake mengubah peristiwa yang ada dalam format JSON berbasis baris ke format Apache ORC](#). ORC adalah format penyimpanan kolumnar yang dioptimalkan untuk pengambilan

data dengan cepat. Peristiwa digabungkan ke dalam penyimpanan data peristiwa, yang merupakan kumpulan peristiwa yang tidak dapat diubah berdasarkan kriteria yang Anda pilih dengan menerapkan pemilih acara [tingkat lanjut](#). Penyeleksi yang Anda terapkan ke penyimpanan data acara mengontrol peristiwa mana yang bertahan dan tersedia untuk Anda kueri. Untuk informasi lebih lanjut tentang CloudTrail Danau, lihat [Bekerja dengan AWS CloudTrail Danau](#) di Panduan AWS CloudTrail Pengguna.

CloudTrail Penyimpanan data acara danau dan kueri menimbulkan biaya. Saat Anda membuat penyimpanan data acara, Anda memilih [opsi harga](#) yang ingin Anda gunakan untuk penyimpanan data acara. Opsi penetapan harga menentukan biaya untuk menelan dan menyimpan peristiwa, dan periode retensi default dan maksimum untuk penyimpanan data acara. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#).

Topik

- [Acara manajemen X-Ray di CloudTrail](#)
- [Peristiwa data X-Ray di CloudTrail](#)
- [Contoh acara X-Ray](#)

Acara manajemen X-Ray di CloudTrail

AWS X-Ray terintegrasi dengan AWS CloudTrail untuk merekam tindakan API yang dibuat oleh pengguna, peran, atau Layanan AWS dalam X-Ray. Anda dapat menggunakan CloudTrail untuk memantau permintaan X-Ray API secara real time dan menyimpan log di Amazon S3, Amazon CloudWatch Log, dan Amazon Events. CloudWatch X-Ray mendukung pencatatan tindakan berikut sebagai peristiwa dalam file CloudTrail log:

Tindakan API yang didukung

- [PutEncryptionConfig](#)
- [GetEncryptionConfig](#)
- [CreateGroup](#)
- [UpdateGroup](#)
- [DeleteGroup](#)
- [GetGroup](#)
- [GetGroups](#)

- [GetInsight](#)
- [GetInsightEvents](#)
- [GetInsightImpactGraph](#)
- [GetInsightSummaries](#)
- [GetSamplingStatisticSummaries](#)

Peristiwa data X-Ray di CloudTrail

[Peristiwa data](#) memberikan informasi tentang operasi sumber daya yang dilakukan pada atau di sumber daya (misalnya, [PutTraceSegments](#), yang mengunggah dokumen segmen ke X-Ray).

Ini juga dikenal sebagai operasi bidang data. Peristiwa data seringkali merupakan aktivitas volume tinggi. Secara default, CloudTrail tidak mencatat peristiwa data. Riwayat CloudTrail peristiwa tidak merekam peristiwa data.

Biaya tambahan berlaku untuk peristiwa data. Untuk informasi selengkapnya tentang CloudTrail harga, lihat [AWS CloudTrail Harga](#).

Anda dapat mencatat peristiwa data untuk jenis sumber daya X-Ray menggunakan CloudTrail konsol AWS CLI, atau operasi CloudTrail API. Untuk informasi selengkapnya tentang cara mencatat peristiwa data, lihat [Mencatat peristiwa data dengan AWS Management Console](#) dan [Mencatat peristiwa data dengan AWS Command Line Interface](#) di Panduan AWS CloudTrail Pengguna.

Tabel berikut mencantumkan jenis sumber daya X-Ray yang dapat Anda log peristiwa data. Kolom tipe peristiwa data (konsol) menunjukkan nilai yang akan dipilih dari daftar tipe peristiwa Data di CloudTrail konsol. Kolom nilai `resources.type` menunjukkan **resources.type** nilai, yang akan Anda tentukan saat mengonfigurasi penyeleksi acara lanjutan menggunakan or. AWS CLI CloudTrail APIs CloudTrailKolom Data yang APIs dicatat ke menampilkan panggilan API yang dicatat CloudTrail untuk jenis sumber daya.

Jenis peristiwa data (konsol)	nilai <code>resources.type</code>	Data APIs masuk ke CloudTrail
Jejak X-Ray	<code>AWS::XRay::Trace</code>	• PutTraceSegments • GetTraceSummaries • GetTraceGraph • GetServiceGraph

Jenis peristiwa data (konsol)	nilai resources.type	Data APIs masuk ke CloudTrail
		• BatchGetTraces • GetTimeSeriesServiceStatistics • PutTelemetryRecords • GetSamplingTargets

Anda dapat mengonfigurasi pemilih acara lanjutan untuk memfilter pada eventName dan readOnly bidang untuk mencatat hanya peristiwa yang penting bagi Anda. Namun, Anda tidak dapat memilih acara dengan menambahkan pemilih resources .ARN bidang, karena jejak X-Ray tidak memiliki ARNs. Untuk informasi selengkapnya tentang bidang ini, lihat [AdvancedFieldSelector](#) di Referensi API AWS CloudTrail . Berikut ini adalah contoh cara menjalankan [put-event-selectors](#) AWS CLI perintah untuk mencatat peristiwa data pada CloudTrail jejak. Anda harus menjalankan perintah di atau menentukan Wilayah tempat jejak dibuat; jika tidak, operasi mengembalikan InvalidHomeRegionException pengecualian.

```
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"
    }
  ]
}'
```

Contoh acara X-Ray

Contoh acara manajemen, **GetEncryptionConfig**

Berikut ini adalah contoh X-Ray GetEncryptionConfig masuk masuk CloudTrail.

Example

```
{
  "eventVersion"=>"1.05",
  "userIdentity"=>{
    "type"=>"AssumedRole",
    "principalId"=>"AR0AJVHBZWD3DN6CI2MHM:MyName",
    "arn"=>"arn:aws:sts::123456789012:assumed-role/MyRole/MyName",
    "accountId"=>"123456789012",
    "accessKeyId"=>"AKIAIOSFODNN7EXAMPLE",
    "sessionContext"=>{
      "attributes"=>{
        "mfaAuthenticated"=>"false",
        "creationDate"=>"2023-7-01T00:24:36Z"
      },
      "sessionIssuer"=>{
        "type"=>"Role",
        "principalId"=>"AR0AJVHBZWD3DN6CI2MHM",
        "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"
}
```

Contoh peristiwa data, **PutTraceSegments**

Berikut ini adalah contoh X-Ray PutTraceSegments entri log peristiwa data masuk 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 Integrasi dengan X-Ray

AWS X-Ray terintegrasi dengan [CloudWatch Application Signals](#), CloudWatch RUM, dan CloudWatch Synthetics untuk memudahkan pemantauan kesehatan aplikasi Anda. Aktifkan aplikasi Anda untuk Sinyal Aplikasi untuk memantau dan memecahkan masalah kesehatan operasional layanan, halaman klien, kenari Synthetics, dan dependensi layanan Anda.

Dengan mengkorelasikan CloudWatch metrik, log, dan jejak X-Ray, peta jejak X-Ray memberikan end-to-end tampilan layanan Anda untuk membantu Anda dengan cepat menentukan kemacetan kinerja dan mengidentifikasi pengguna yang terkena dampak.

Dengan CloudWatch RUM, Anda dapat melakukan pemantauan pengguna nyata untuk mengumpulkan dan melihat data sisi klien tentang kinerja aplikasi web Anda dari sesi pengguna yang sebenarnya dalam waktu dekat. Dengan AWS X-Ray dan CloudWatch RUM, Anda dapat menganalisis dan men-debug jalur permintaan mulai dari pengguna akhir aplikasi Anda melalui layanan AWS terkelola hilir. Ini akan membantu Anda untuk mengidentifikasi tren latensi dan kesalahan yang berdampak pada pengguna akhir Anda.

Topik

- [CloudWatch RUM dan AWS X-Ray](#)
- [Debugging kenari CloudWatch sintetis menggunakan X-Ray](#)

CloudWatch RUM dan AWS X-Ray

Dengan Amazon CloudWatch RUM, Anda dapat melakukan pemantauan pengguna nyata untuk mengumpulkan dan melihat data sisi klien tentang kinerja aplikasi web Anda dari sesi pengguna aktual dalam waktu hampir nyata. Dengan AWS X-Ray dan CloudWatch RUM, Anda dapat menganalisis dan men-debug jalur permintaan mulai dari pengguna akhir aplikasi Anda melalui layanan AWS terkelola hilir. Ini akan membantu Anda untuk mengidentifikasi tren latensi dan kesalahan yang berdampak pada pengguna akhir Anda.

Setelah Anda mengaktifkan penelusuran X-Ray sesi pengguna, CloudWatch RUM menambahkan header jejak X-Ray ke permintaan HTTP yang diizinkan, dan merekam segmen X-Ray untuk permintaan HTTP yang diizinkan. Anda kemudian dapat melihat jejak dan segmen dari sesi pengguna ini di X-Ray dan CloudWatch konsol, termasuk peta jejak X-Ray.

Note

CloudWatch RUM tidak terintegrasi dengan aturan pengambilan sampel X-Ray. Sebaliknya, pilih persentase sampling ketika Anda mengatur aplikasi Anda untuk menggunakan CloudWatch RUM. Jejak yang dikirim dari CloudWatch RUM mungkin dikenakan biaya tambahan. Untuk informasi selengkapnya, lihat [harga AWS X-Ray](#).

Secara default, jejak sisi klien yang dikirim dari CloudWatch RUM tidak terhubung ke jejak sisi server. Untuk menghubungkan jejak sisi klien dengan jejak sisi server, konfigurasi klien web CloudWatch RUM untuk menambahkan header jejak X-Ray ke permintaan HTTP ini.

Warning

Mengkonfigurasi klien web CloudWatch RUM untuk menambahkan header jejak X-Ray ke permintaan HTTP dapat menyebabkan berbagi sumber daya lintas asal (CORS) gagal. Untuk menghindari hal ini, tambahkan header `X-Amzn-Trace-Id` HTTP ke daftar header yang diizinkan pada konfigurasi CORS layanan hilir Anda. Jika Anda menggunakan API Gateway sebagai hilir, lihat [Mengaktifkan CORS untuk sumber daya REST API](#). Kami sangat menyarankan Anda menguji aplikasi Anda sebelum menambahkan header jejak sinar X sisi klien di lingkungan produksi. Untuk informasi selengkapnya, lihat [dokumentasi klien web CloudWatch RUM](#).

Untuk informasi selengkapnya tentang pemantauan pengguna nyata di CloudWatch, lihat [Menggunakan CloudWatch RUM](#). Untuk mengatur aplikasi Anda agar menggunakan CloudWatch RUM, termasuk melacak sesi pengguna dengan X-Ray, lihat [Mengatur aplikasi untuk menggunakan CloudWatch RUM](#).

Debugging kenari CloudWatch sintetis menggunakan X-Ray

CloudWatch Synthetics adalah layanan yang dikelola sepenuhnya yang memungkinkan Anda memantau titik akhir Anda dan APIs menggunakan kenari skrip yang berjalan 24 jam per hari, sekali per menit.

Anda dapat menyesuaikan penulisan canary untuk memeriksa perubahan dalam:

- Ketersediaan
- Latensi
- Transaksi
- Tautan yang rusak atau mati
- Step-by-step penyelesaian tugas
- Kesalahan memuat halaman
- Latensi Beban untuk aset UI
- Arus wizard kompleks
- Aliran checkout dalam aplikasi Anda

Canary mengikuti rute yang sama dan melakukan tindakan dan perilaku yang sama seperti pelanggan Anda, dan terus-menerus memverifikasi pengalaman pelanggan.

Untuk mempelajari selengkapnya tentang cara mengonfigurasi tes Synthetics, lihat [Menggunakan Synthetics untuk Membuat dan Mengelola Canary](#).



Contoh berikut menunjukkan kasus penggunaan umum untuk debugging masalah yang meningkatkan Synthetics canary Anda. Setiap contoh menunjukkan strategi kunci untuk debugging menggunakan peta jejak atau konsol X-Ray Analytics.

Untuk informasi selengkapnya tentang cara membaca dan berinteraksi dengan peta jejak, lihat [Melihat Peta Layanan](#).

Untuk informasi selengkapnya tentang cara membaca dan berinteraksi dengan konsol X-Ray Analytics, lihat [Berinteraksi dengan Konsol AWS X-Ray Analytics](#).

Topik

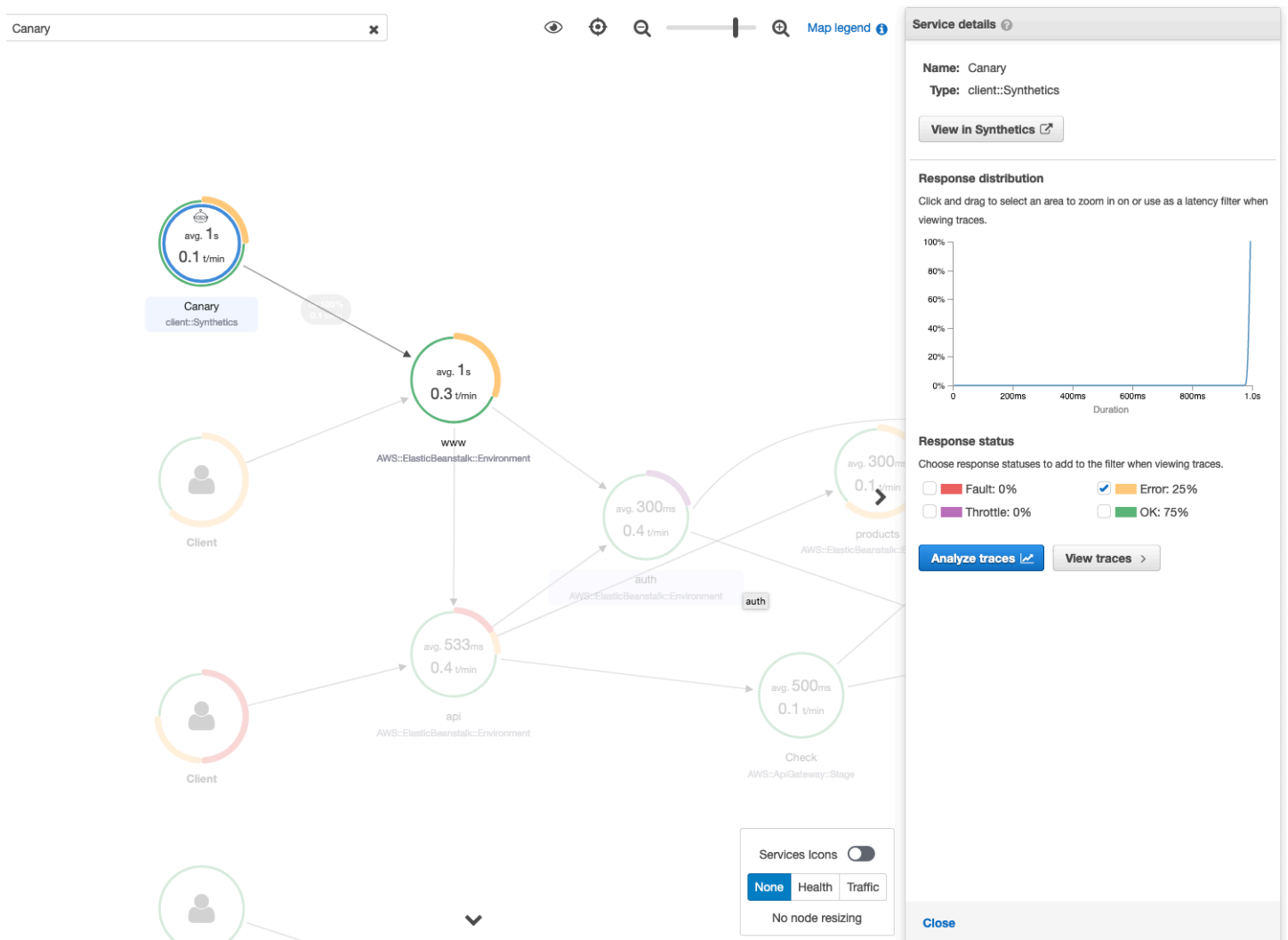
- [Lihat kenari dengan peningkatan pelaporan kesalahan di peta jejak](#)
- [Gunakan peta detail jejak untuk setiap jejak untuk melihat setiap permintaan secara detail](#)

- [Tentukan akar masalah kegagalan yang sedang berlangsung di layanan hulu dan hilir](#)
- [Identifikasi bottleneck dan tren performa](#)
- [Bandingkan tingkat latensi dan kesalahan atau kesalahan sebelum dan sesudah perubahan](#)
- [Tentukan cakupan kenari yang diperlukan untuk semua APIs dan URLs](#)
- [Gunakan grup untuk fokus pada uji synthetics](#)

Lihat kenari dengan peningkatan pelaporan kesalahan di peta jejak

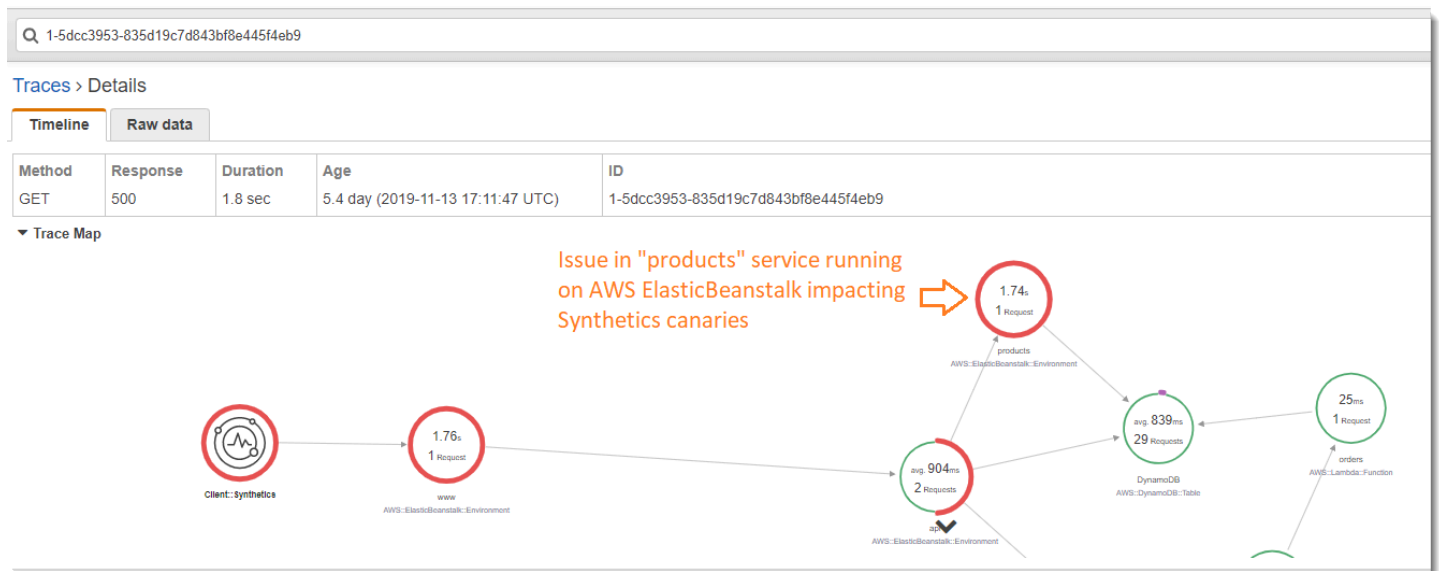
[Untuk melihat kenari mana yang memiliki peningkatan kesalahan, kesalahan, laju pelambatan, atau waktu respons lambat dalam peta jejak X-Ray Anda, Anda dapat menyorot node klien kenari Synthetics menggunakan filter. `Client::Synthetic`](#) Meneklik simpul menampilkan distribusi waktu respons dari seluruh permintaan. Meneklik edge antara dua simpul menunjukkan detail tentang permintaan yang melintasi koneksi itu. Anda juga dapat melihat node yang disimpulkan “jarak jauh” untuk layanan hilir terkait di peta jejak Anda.

Ketika Anda mengeklik simpul Synthetics, ada Lihat dalam Synthetics pada panel samping yang mengalihkan Anda ke konsol Synthetics tempat Anda dapat memeriksa detail canary.



Gunakan peta detail jejak untuk setiap jejak untuk melihat setiap permintaan secara detail

Untuk menentukan layanan mana yang menghasilkan latensi paling banyak atau menyebabkan kesalahan, panggil peta detail jejak dengan memilih jejak di peta jejak. Peta detail jejak individu menampilkan end-to-end jalur permintaan tunggal. Gunakan ini untuk memahami layanan yang dipanggil, dan memvisualisasikan layanan hulu dan hilir.



Tentukan akar masalah kegagalan yang sedang berlangsung di layanan hulu dan hilir

Setelah Anda menerima CloudWatch alarm untuk kegagalan dalam kenari Synthetics, gunakan pemodelan statistik pada data jejak di X-Ray untuk menentukan kemungkinan akar penyebab masalah dalam konsol X-Ray Analytics. Di konsol Analytics, tabel Akar Penyebab Waktu Respons menunjukkan jalur entitas yang dicatat. X-Ray menentukan jalur mana yang ada di pelacakan Anda adalah penyebab paling mungkin untuk waktu respons. Format menunjukkan hierarki entitas yang ditemui, berakhir dengan akar masalah waktu respons.

Contoh berikut menunjukkan bahwa tes Synthetics untuk API “XXX” berjalan pada API Gateway gagal karena pengecualian kapasitas throughput dari tabel Amazon DynamoDB.

Canary

Client

avg. 1.2s
1 t/min

Canary
client::Synthetics

avg. 900ms
2 t/min

www
AWS::ElasticBeanstalk::Environment

avg. 533ms
4 t/min

api
AWS::ElasticBeanstalk::Environment

avg. 300ms
4 t/min

auth
AWS::ElasticBeanstalk::Environment

Fault 67%
1 t/min

Client

Select the node

Select to view faults and analyze traces

Services Icons

NoneHealthTraffic

No node resizing

Service details

Name: Canary

Type: client::Synthetics

View In Synthetics

Response distribution

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

100%

80%

60%

40%

20%

0%

0

200ms

400ms

600ms

800ms

1.0s

1.2s

Duration

Response status

Choose response statuses to add to the filter when viewing traces.

Fault: 67%

Error: 0%

Throttle: 0%

OK: 33%

Analyze traces

View traces

Close

Service map

Traces

Analytics

Configuration

Sampling

Encryption

FAULT ROOT CAUSE

www (AWS::ElasticBeanstalk::Environment) → error ⇒ api (AWS::ElasticBeanstalk::Environment) → error ⇒ products (AWS::ElasticBeanstalk::Environment) → error ⇒ products (AWS::DynamoDB::Table)

COUNT

%

4

100.00%

FAULT ROOT CAUSE MESSAGE

ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API. status code: 4

COUNT

%

4

100.00%

Root cause analysis indicating throughput capacity exceeded for DynamoDB table

AWS:CANARY_ARN	COUNT
arn:aws:synthetics:us-east-1:779168132807:canary:www-test	118

Annotation.aws.canary_arn

Annotation.acl_cached

Annotation.authenticated

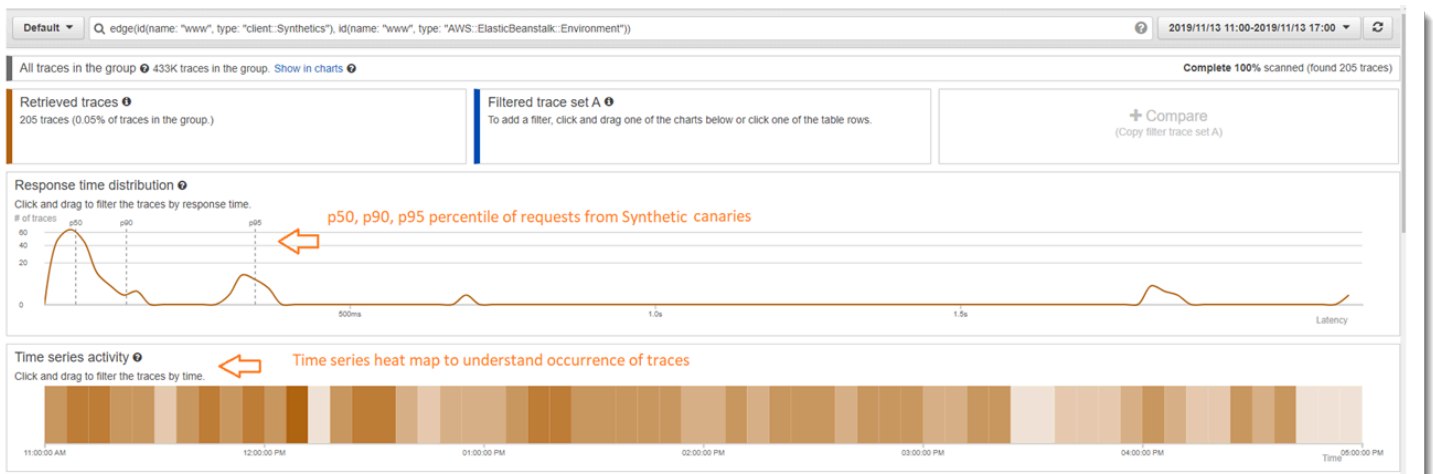
Annotation.cold_start

Annotation.credentials_cached

Annotation.queries

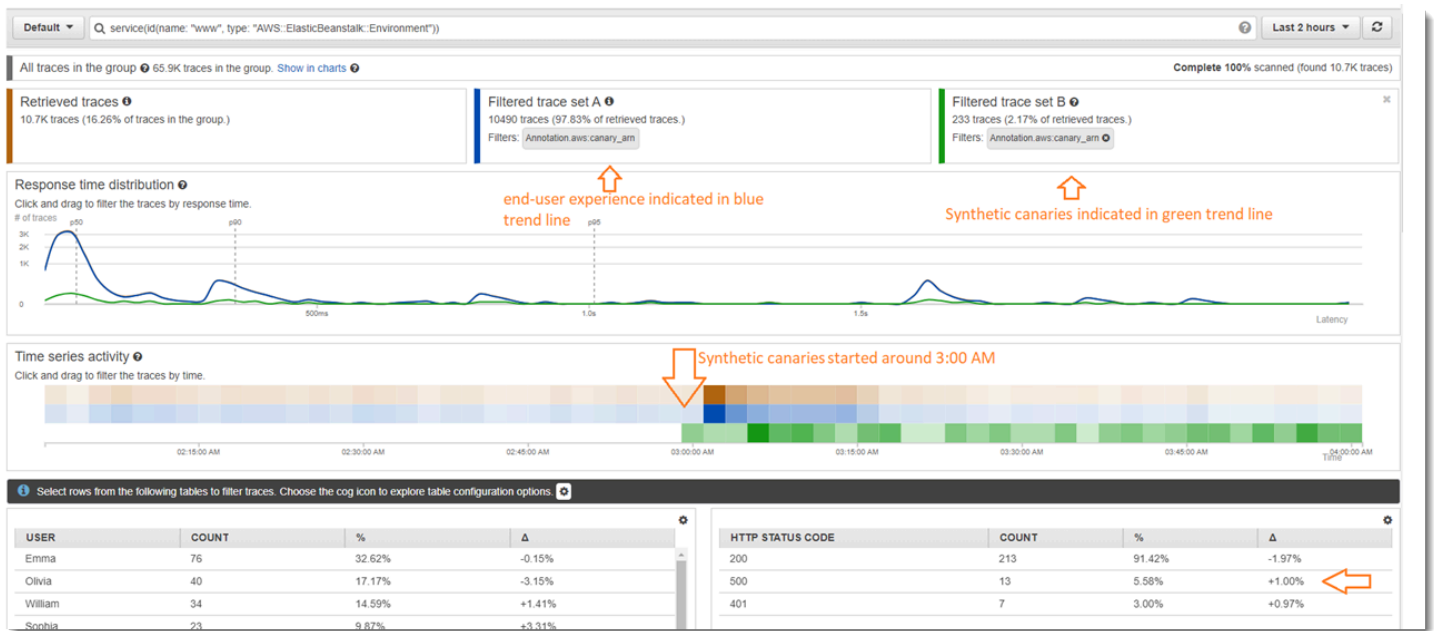
Identifikasi bottleneck dan tren performa

Anda dapat melihat tren kinerja titik akhir Anda dari waktu ke waktu menggunakan lalu lintas berkelanjutan dari kenari Synthetics Anda untuk mengisi peta detail jejak selama periode waktu tertentu.



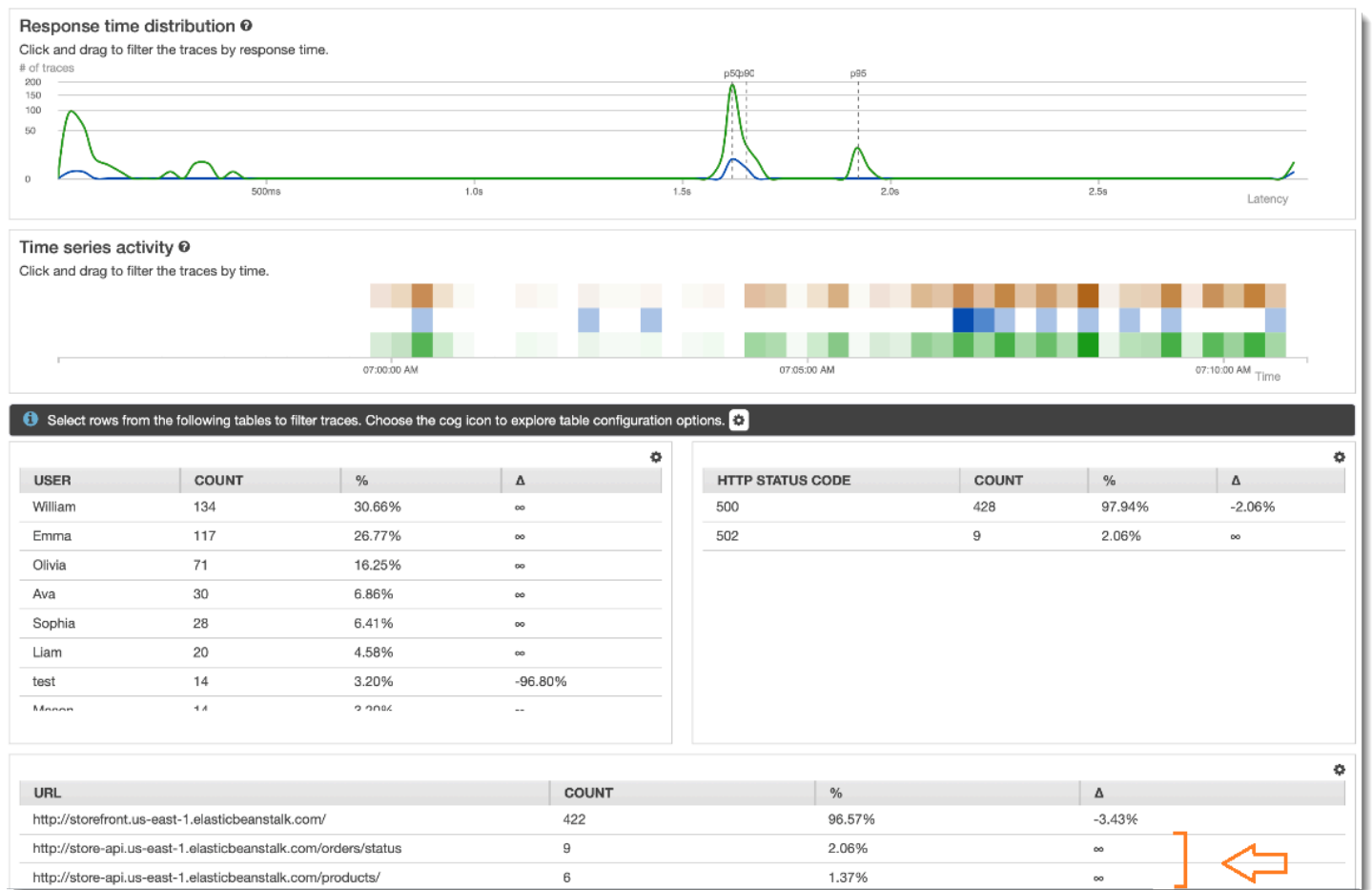
Bandingkan tingkat latensi dan kesalahan atau kesalahan sebelum dan sesudah perubahan

Pinpoint waktu perubahan terjadi untuk menghubungkan perubahan itu dengan peningkatan masalah yang ditangkap oleh kenari Anda. Gunakan konsol Analitik X-Ray untuk menentukan rentang waktu sebelum dan sesudah setelah pelacakan yang berbeda ditetapkan, membuat diferensiasi visual dalam distribusi waktu respons.



Tentukan cakupan kenari yang diperlukan untuk semua APIs dan URLs

Gunakan Analitik X-Ray untuk membandingkan pengalaman canary dengan pengguna. UI di bawah ini menunjukkan garis tren biru untuk canary dan garis hijau untuk pengguna. Anda juga dapat mengidentifikasi bahwa dua dari ketiganya URLs tidak memiliki tes kenari.

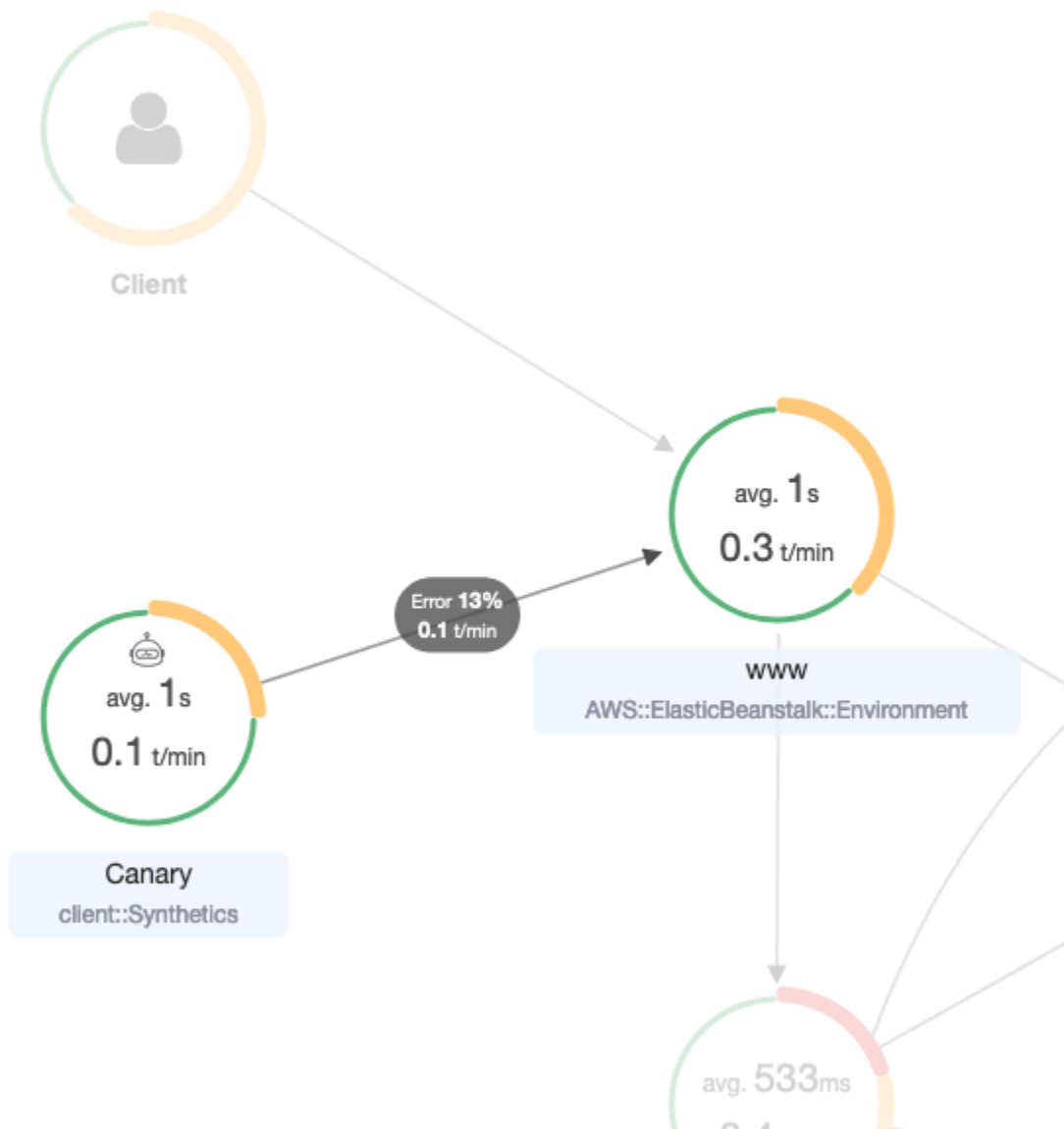


Gunakan grup untuk fokus pada uji synthetics

Anda dapat membuat grup X-Ray menggunakan ekspresi filter untuk fokus pada serangkaian alur kerja tertentu, seperti tes Synthetics untuk aplikasi “www” yang berjalan di AWS Elastic Beanstalk. Gunakan [kata kunci kompleks](#) `service()` dan `edge()` untuk memfilter melalui layanan dan edge.

Example Ekspresi filter grup

```
"edge(id(name: "www", type: "client::Synthetics"), id(name: "www", type: "AWS::ElasticBeanstalk::Environment"))"
```



Melacak perubahan konfigurasi enkripsi X-Ray AWS Config

AWS X-Ray terintegrasi dengan AWS Config untuk merekam perubahan konfigurasi yang dibuat pada sumber daya enkripsi X-Ray Anda. Anda dapat menggunakan AWS Config inventaris sumber daya enkripsi X-Ray, mengaudit riwayat konfigurasi X-Ray, dan mengirim pemberitahuan berdasarkan perubahan sumber daya.

AWS Config mendukung pencatatan perubahan sumber daya enkripsi X-Ray berikut sebagai peristiwa:

- Perubahan konfigurasi – Mengubah atau menambahkan kunci enkripsi, atau kembali ke pengaturan enkripsi X-Ray default.

Gunakan petunjuk berikut untuk mempelajari cara membuat koneksi dasar antara X-Ray dan AWS Config.

Membuat pemicu fungsi Lambda

Anda harus memiliki ARN dari AWS Lambda fungsi kustom sebelum Anda dapat membuat aturan khusus AWS Config . Ikuti petunjuk ini untuk membuat fungsi dasar dengan Node.js yang mengembalikan nilai kepatuhan atau ketidakpatuhan kembali ke AWS Config berdasarkan status sumber daya XrayEncryptionConfig.

Untuk membuat fungsi Lambda dengan pemicu perubahan AWS::Xray EncryptionConfig

1. Buka [Konsol Lambda](#). Pilih Buat fungsi.
2. Pilih Blueprints, lalu filter pustaka cetak biru untuk cetak biru. config-rule-change-triggered Klik tautan di nama cetak biru atau pilih Konfigurasi untuk melanjutkan.
3. Tentukan bidang berikut untuk mengonfigurasi cetak biru:
 - Untuk Name, ketik nama.
 - Untuk Peran, pilih Buat peran baru dari templat.
 - Untuk Nama peran, ketik nama.
 - Untuk Templat Kebijakan, pilih Izin aturan AWS Config .
4. Pilih Buat fungsi untuk membuat dan menampilkan fungsi Anda di AWS Lambda konsol.
5. Edit kode fungsi Anda untuk mengganti AWS::EC2::Instance dengan AWS::XrayEncryptionConfig. Anda juga dapat memperbarui bidang deskripsi untuk mencerminkan perubahan ini.

Kode Default

```
if (configurationItem.resourceType !== 'AWS::EC2::Instance') {
    return 'NOT_APPLICABLE';
} else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
    return 'COMPLIANT';
}
```

```
return 'NON_COMPLIANT';
```

Kode yang Diperbarui

```
if (configurationItem.resourceType !== 'AWS::XRay::EncryptionConfig') {
    return 'NOT_APPLICABLE';
} else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
    return 'COMPLIANT';
}
return 'NON_COMPLIANT';
```

6. Tambahkan yang berikut ke peran eksekusi Anda di IAM untuk akses ke X-Ray. Izin ini mengizinkan akses hanya-baca ke sumber daya X-Ray Anda. Kegagalan untuk menyediakan akses ke sumber daya yang sesuai akan mengakibatkan pesan di luar cakupan AWS Config sejak mengevaluasi fungsi Lambda yang terkait dengan aturan.

```
{
  "Sid": "Stmt1529350291539",
  "Action": [
    "xray:GetEncryptionConfig"
  ],
  "Effect": "Allow",
  "Resource": "*"
}
```

Membuat AWS Config aturan khusus untuk x-ray

Saat fungsi Lambda dibuat, perhatikan ARN fungsi tersebut, dan buka AWS Config konsol untuk membuat aturan kustom Anda.

Untuk membuat AWS Config aturan untuk X-Ray

1. Buka halaman [Aturan dari konsol AWS Config tersebut](#).
2. Pilih Tambahkan aturan, lalu pilih Tambahkan aturan kustom.
3. Di Fungsi ARN AWS Lambda , masukkan ARN yang terkait dengan fungsi Lambda yang ingin Anda gunakan.
4. Pilih tipe pemicu untuk mengatur:

- Perubahan konfigurasi — AWS Config memicu evaluasi ketika sumber daya apa pun yang cocok dengan cakupan aturan berubah dalam konfigurasi. Evaluasi berjalan setelah AWS Config mengirimkan pemberitahuan perubahan item konfigurasi.
- Berkala — AWS Config menjalankan evaluasi untuk aturan pada frekuensi yang Anda pilih (misalnya, setiap 24 jam).

5. Untuk jenis Sumber Daya, pilih EncryptionConfigdi bagian X-Ray.

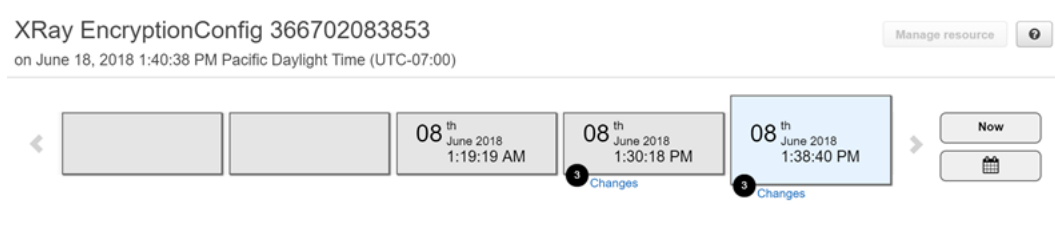
6. Pilih Simpan.

AWS Config Konsol mulai mengevaluasi kepatuhan aturan segera. Evaluasi ini dapat memakan waktu beberapa menit hingga selesai.

Sekarang aturan ini sesuai, AWS Config dapat mulai menyusun riwayat audit. AWS Config mencatat perubahan sumber daya dalam bentuk garis waktu. Untuk setiap perubahan dalam timeline peristiwa, AWS Config buat tabel dalam format dari/ke untuk menunjukkan apa yang berubah dalam representasi JSON dari kunci enkripsi. Dua perubahan bidang yang terkait dengan EncryptionConfig adalah `Configuration.type` dan `Configuration.keyID`.

Contoh hasil

Berikut ini adalah contoh AWS Config timeline yang menunjukkan perubahan yang dibuat pada tanggal dan waktu tertentu.



Berikut ini adalah contoh entri AWS Config perubahan. Dari/ke format menggambarkan apa yang berubah. Contoh ini menunjukkan bahwa pengaturan enkripsi X-Ray default diubah ke kunci enkripsi yang ditentukan.

▼ Changes **3**

Configuration Changes **3**

Field	From	To
SupplementaryConfiguration.unsupportedResources		- Array [1] - Object - resourceId: "arn:aws:kms:us-west-2:366702083853:key/e0531084-06ad-4d7f-9ea6-53dd693a945c" - resourceType: "AWS::KMS::Key"
Configuration.type	"NONE"	"KMS"
Configuration.keyId		"arn:aws:kms:us-west-2:366702083853:key/e0531084-06ad-4d7f-9ea6-53dd693a945c"

Notifikasi Amazon SNS

Untuk diberi tahu tentang perubahan konfigurasi, atur AWS Config untuk mempublikasikan notifikasi Amazon SNS. Untuk informasi selengkapnya, lihat [Memantau Perubahan AWS Config Sumber Daya melalui Email](#).

Amazon Elastic Compute Cloud dan AWS X-Ray

Anda dapat menginstal dan menjalankan daemon X-Ray pada EC2 instance Amazon dengan skrip data pengguna. Lihat [Menjalankan daemon X-Ray di Amazon EC2](#) untuk instruksi.

Gunakan profil instans untuk memberikan izin daemon untuk mengunggah data penelusuran ke X-Ray. Untuk informasi selengkapnya, lihat [Memberikan izin kepada daemon untuk mengirim data ke X-Ray](#).

AWS Elastic Beanstalk dan AWS X-Ray

AWS Elastic Beanstalk Platform termasuk daemon X-Ray. Anda dapat [menjalankan daemon](#) dengan mengatur opsi di konsol Elastic Beanstalk atau dengan file konfigurasi.

Pada platform Java SE, Anda dapat menggunakan file Buildfile untuk membangun aplikasi Anda dengan Maven atau Gradle pada instans. X-Ray SDK for Java AWS SDK untuk Java dan tersedia dari Maven, sehingga Anda hanya dapat menerapkan kode aplikasi dan membangun on-instance untuk menghindari bundling dan mengunggah semua dependensi Anda.

Anda dapat menggunakan properti lingkungan Elastic Beanstalk untuk mengonfigurasi SDK X-Ray. Metode yang digunakan Elastic Beanstalk untuk melalui properti lingkungan untuk aplikasi Anda bervariasi berdasarkan platform. Gunakan variabel lingkungan atau properti sistem X-Ray SDK tergantung pada platform Anda.

- [Platform Node.js](#) – Gunakan [Variabel lingkungan](#)

- [Platform Java SE](#) – Gunakan [Variabel lingkungan](#)
- [Platform Tomcat](#) – Gunakan [Properti sistem](#)

Untuk informasi selengkapnya, lihat [Mengonfigurasi AWS X-Ray Debugging](#) di Panduan AWS Elastic Beanstalk Pengembang.

Elastic Load Balancing dan AWS X-Ray

Elastic Load Balancing aplikasi penyeimbang beban menambahkan ID penelusuran untuk permintaan HTTP masuk di header bernama X-Amzn-Trace-Id.

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

Format ID jejak X-Ray

trace_idX-Ray terdiri dari tiga angka yang dipisahkan oleh tanda hubung. Contohnya, 1-58406520-a006649127e371903a2de979. Hal ini mencakup:

- Nomor versi, yaitu 1.
- Waktu permintaan asli dalam waktu epoch Unix menggunakan 8 digit heksadesimal.

Misalnya, 10:00 AM 1 Desember 2016 PST dalam waktu epoch adalah 1480615200 detik atau 58406520 dalam digit heksadesimal.

- Pengidentifikasi 96-bit yang unik secara global untuk jejak dalam 24 digit heksadesimal.

Penyeimbang beban tidak mengirim data ke X-Ray, dan tidak muncul sebagai simpul pada peta layanan Anda.

Untuk informasi selengkapnya, lihat [Meminta Penelusuran untuk Application Load Balancer Anda](#) dalam Panduan Developer Elastic Load Balancing.

Amazon EventBridge dan AWS X-Ray

AWS X-Ray terintegrasi dengan Amazon EventBridge untuk melacak peristiwa yang dilewati EventBridge. [Jika layanan yang diinstrumentasi dengan X-Ray SDK mengirimkan peristiwa ke EventBridge, konteks penelusuran akan disebarkan ke target peristiwa hilir dalam header penelusuran.](#) SDK X-Ray secara otomatis mengambil header penelusuran dan menerapkannya ke

instrumentasi berikutnya. Kontinuitas ini memungkinkan pengguna untuk menelusuri, menganalisis, dan men-debug seluruh layanan downstream dan memberikan tampilan yang lebih lengkap dari sistem mereka.

Untuk informasi selengkapnya, lihat [Integrasi EventBridge X-Ray](#) di Panduan EventBridge Pengguna.

Melihat sumber dan target pada peta layanan X-Ray

[Peta jejak](#) X-Ray menampilkan node EventBridge peristiwa yang menghubungkan layanan sumber dan target, seperti pada contoh berikut:



Sebarkan konteks penelusuran untuk target peristiwa

X-Ray SDK memungkinkan sumber EventBridge peristiwa untuk menyebarkan konteks jejak ke target peristiwa hilir. Contoh khusus bahasa berikut menunjukkan pemanggilan EventBridge dari fungsi Lambda tempat penelusuran [aktif](#) diaktifkan:

Java

Tambahkan dependensi yang diperlukan untuk X-Ray:

- [AWS X-Ray SDK for Java](#)
- [AWS X-Ray Perekam SDK for Java](#)

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
```

```

import com.amazonaws.xray.AWSXRay;
import com.amazonaws.services.eventbridge.AmazonEventBridge;
import com.amazonaws.services.eventbridge.AmazonEventBridgeClientBuilder;
import com.amazonaws.services.eventbridge.model.PutEventsRequest;
import com.amazonaws.services.eventbridge.model.PutEventsRequestEntry;
import com.amazonaws.services.eventbridge.model.PutEventsResult;
import com.amazonaws.services.eventbridge.model.PutEventsResultEntry;
import com.amazonaws.xray.handlers.TracingHandler;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.lang.StringBuilder;
import java.util.Map;
import java.util.List;
import java.util.Date;
import java.util.Collections;

/*
    Add the necessary dependencies for XRay:
    https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-xray
    https://mvnrepository.com/artifact/com.amazonaws/aws-xray-recorder-sdk-aws-sdk
*/
public class Handler implements RequestHandler<SQSEvent, String>{
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);

    /*
        build EventBridge client
    */
    private static final AmazonEventBridge eventsClient =
    AmazonEventBridgeClientBuilder
        .standard()
        // instrument the EventBridge client with the XRay Tracing Handler.
        // the AWSXRay globalRecorder will retrieve the tracing-context
        // from the lambda function and inject it into the HTTP header.
        // be sure to enable 'active tracing' on the lambda function.
        .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
        .build();

    @Override
    public String handleRequest(SQSEvent event, Context context)
    {
        PutEventsRequestEntry putEventsRequestEntry0 = new PutEventsRequestEntry();
        putEventsRequestEntry0.setTime(new Date());
    }

```

```

putEventsRequestEntry0.setSource("my-lambda-function");
putEventsRequestEntry0.setDetailType("my-lambda-event");
putEventsRequestEntry0.setDetail("{\"lambda-source\":\"sqs\"}");
PutEventsRequest putEventsRequest = new PutEventsRequest();
putEventsRequest.setEntries(Collections.singletonList(putEventsRequestEntry0));
// send the event(s) to EventBridge
PutEventsResult putEventsResult = eventsClient.putEvents(putEventsRequest);
try {
    logger.info("Put Events Result: {}", putEventsResult);
} catch (Exception e) {
    e.printStackTrace();
}
return "success";
}
}

```

Python

Tambahkan dependensi berikut ke file requirements.txt Anda:

```
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#

Tambahkan paket X-Ray berikut untuk dependensi C # Anda:

```
<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 dan AWS X-Ray

Anda dapat menggunakan AWS X-Ray untuk melacak AWS Lambda fungsi Anda. Lambda menjalankan [daemon X-Ray](#) dan merekam segmen dengan detail tentang memanggil dan menjalankan fungsi. Untuk instrumentasi selengkapya, Anda dapat memaketkan SDK X-Ray dengan fungsi Anda untuk mencatat panggilan keluar dan menambahkan anotasi dan metadata.

Jika fungsi Lambda Anda disebut oleh layanan instrumen lain, Lambda menelusuri permintaan yang telah dijadikan sampel tanpa konfigurasi tambahan. Layanan upstream dapat berupa aplikasi web

berinstrumen atau fungsi Lambda lain. Layanan Anda dapat menjalankan fungsi secara langsung dengan klien AWS SDK yang diinstrumentasi, atau dengan memanggil API Gateway API dengan klien HTTP yang diinstrumentasi.

AWS X-Ray mendukung penelusuran aplikasi berbasis peristiwa menggunakan dan Amazon AWS Lambda SQS. Gunakan CloudWatch konsol untuk melihat tampilan terhubung dari setiap permintaan saat antri dengan Amazon SQS dan diproses oleh fungsi Lambda hilir. Jejak dari produsen pesan hulu secara otomatis ditautkan ke jejak dari node konsumen Lambda hilir, menciptakan end-to-end tampilan aplikasi. Untuk informasi selengkapnya, lihat [melacak aplikasi berbasis peristiwa](#).

Note

Jika Anda mengaktifkan jejak untuk fungsi Lambda hilir, Anda juga harus mengaktifkan jejak untuk fungsi Lambda root yang memanggil fungsi hilir agar fungsi hilir menghasilkan jejak.

Jika fungsi Lambda Anda berjalan sesuai jadwal, atau dipanggil oleh layanan yang tidak diinstrumentasi, Anda dapat mengonfigurasi Lambda untuk sampel dan mencatat panggilan dengan penelusuran aktif.

Untuk mengkonfigurasi integrasi X-Ray pada suatu AWS Lambda fungsi

1. Buka [konsol AWS Lambda](#).
2. Pilih Fungsi dari bar navigasi kiri.
3. Pilih fungsi Anda.
4. Pada tab Konfigurasi, gulir ke bawah ke kartu alat pemantauan tambahan. Anda juga dapat menemukan kartu ini dengan memilih alat Pemantauan dan operasi di panel navigasi kiri.
5. Pilih Edit.
6. Di bagian X-Ray AWS , aktifkan Penelusuran aktif.

Saat waktu aktif dengan SDK X-Ray yang sesuai, Lambda juga menjalankan daemon X-Ray.

X-Ray SDKs pada Lambda

- X-Ray SDK for Go – Go 1.7 dan waktu aktif yang lebih baru
- X-Ray SDK for Java – Waktu aktif Java 8
- X-Ray SDK untuk Node.js – Node.js 4.3 dan waktu aktif yang lebih baru

- X-Ray SDK for Python – Python 2.7, Python 3.6, dan waktu aktif yang lebih baru
- X-Ray SDK for .NET – .NET Core 2.0 dan waktu aktif yang lebih baru

Untuk menggunakan SDK X-Ray pada Lambda, paketkan dengan kode fungsi Anda setiap kali Anda membuat versi baru. Anda dapat menginstrumen fungsi Lambda Anda dengan metode yang sama yang Anda gunakan untuk aplikasi instrumen yang berjalan pada layanan lain. Perbedaan utamanya adalah Anda tidak menggunakan SDK untuk instrumen permintaan masuk, membuat keputusan pengambilan sampel, dan membuat segmen.

Perbedaan lain antara menginstrumentasi fungsi Lambda dan aplikasi web adalah bahwa segmen yang dibuat dan dikirim Lambda ke X-Ray tidak dapat dimodifikasi oleh kode fungsi Anda. Anda dapat membuat subsegmen dan mencatat anotasi dan metadata di dalamnya, namun Anda tidak dapat menambahkan anotasi dan metadata ke segmen induk.

Untuk informasi selengkapnya, lihat [Menggunakan X-Ray AWS](#) di Panduan Developer AWS Lambda

Amazon SNS dan AWS X-Ray

[Anda dapat menggunakan AWS X-Ray dengan Amazon Simple Notification Service \(Amazon SNS\) untuk melacak dan menganalisis permintaan saat mereka melakukan perjalanan melalui topik SNS Anda ke layanan berlangganan yang didukung SNS Anda.](#) Gunakan penelusuran X-Ray dengan Amazon SNS untuk menganalisis latensi dalam pesan Anda dan layanan back-end mereka, seperti berapa lama permintaan dihabiskan dalam suatu topik, dan berapa lama waktu yang dibutuhkan untuk menyampaikan pesan ke setiap langganan topik. Amazon SNS mendukung penelusuran X-Ray untuk topik standar dan FIFO.

Jika Anda mempublikasikan ke topik Amazon SNS dari layanan yang sudah diinstrumentasi dengan X-Ray, Amazon SNS meneruskan konteks jejak dari penerbit ke pelanggan. Selain itu, Anda dapat mengaktifkan penelusuran aktif untuk mengirim data segmen tentang langganan Amazon SNS Anda ke X-Ray untuk pesan yang diterbitkan dari klien SNS yang diinstrumentasi. [Aktifkan penelusuran aktif](#) untuk topik Amazon SNS dengan menggunakan konsol Amazon SNS, atau dengan menggunakan Amazon SNS API atau CLI. Lihat [Instrumentasi aplikasi Anda](#) untuk informasi selengkapnya tentang menginstrumentasi klien SNS Anda.

Konfigurasi penelusuran aktif Amazon SNS

Anda dapat menggunakan konsol Amazon SNS atau AWS CLI atau SDK untuk mengonfigurasi penelusuran aktif Amazon SNS.

Saat Anda menggunakan konsol Amazon SNS, Amazon SNS mencoba membuat izin yang diperlukan agar SNS memanggil X-Ray. Upaya dapat ditolak jika Anda tidak memiliki izin yang cukup untuk mengubah kebijakan sumber daya X-Ray. Untuk informasi selengkapnya tentang izin ini, lihat [Identitas dan manajemen akses di Amazon SNS](#) dan [Contoh kasus untuk kontrol akses Amazon SNS](#) di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon. Untuk informasi selengkapnya tentang mengaktifkan penelusuran aktif menggunakan konsol Amazon SNS, lihat [Mengaktifkan penelusuran aktif pada topik Amazon SNS](#) di Panduan Pengembang Layanan Pemberitahuan Sederhana Amazon.

Saat menggunakan AWS CLI atau SDK untuk mengaktifkan penelusuran aktif, Anda harus mengonfigurasi izin secara manual menggunakan kebijakan berbasis sumber daya. Gunakan [PutResourcePolicy](#) untuk mengonfigurasi X-Ray dengan kebijakan berbasis sumber daya yang diperlukan untuk memungkinkan Amazon SNS mengirim jejak ke X-Ray.

Example Contoh kebijakan berbasis sumber daya X-Ray untuk penelusuran aktif Amazon SNS

Contoh dokumen kebijakan ini menentukan izin yang diperlukan Amazon SNS untuk mengirim data jejak ke 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"
  }
}
]
}

```

Gunakan CLI untuk membuat kebijakan berbasis sumber daya yang memberikan izin Amazon SNS untuk mengirim data jejak ke X-Ray:

```

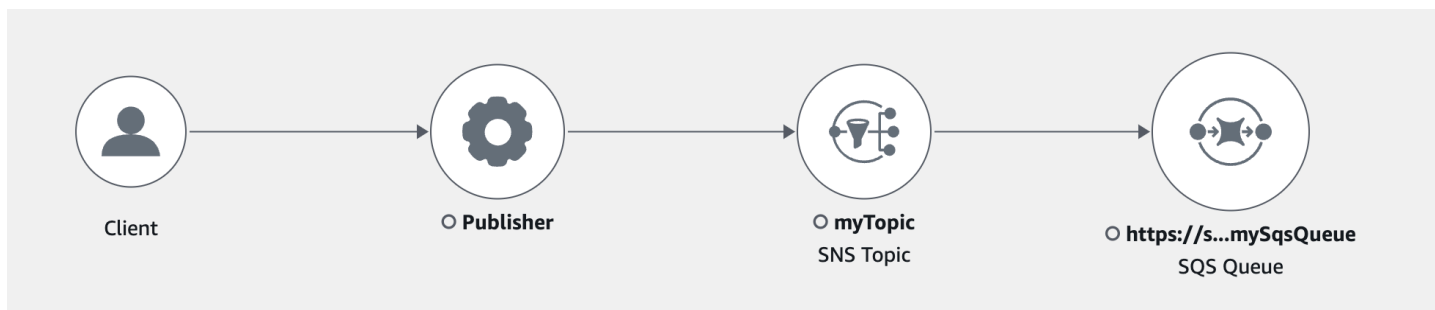
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
'{ "Version": "2012-10-17", "Statement": [ { "Sid": "SNSAccess", "Effect": "Allow",
"Principal": { "Service": "sns.amazonaws.com" }, "Action": [ "xray:PutTraceSegments",
"xray:GetSamplingRules", "xray:GetSamplingTargets" ], "Resource": "*",
"Condition": { "StringEquals": { "aws:SourceAccount": "account-id" }, "StringLike":
{ "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name" } } } ] }'

```

Untuk menggunakan contoh ini, ganti *partition*, *region*, *account-id*, dan *topic-name* dengan AWS partisi tertentu, wilayah, ID akun, dan nama topik Amazon SNS. Untuk memberikan izin kepada semua topik Amazon SNS untuk mengirim data jejak ke X-Ray, ganti nama topik dengan. *

Lihat jejak penerbit dan pelanggan Amazon SNS di konsol X-Ray

Gunakan konsol X-Ray untuk melihat peta jejak dan melacak detail yang menampilkan tampilan penayang dan pelanggan Amazon SNS yang terhubung. Saat penelusuran aktif Amazon SNS diaktifkan untuk suatu topik, peta jejak X-Ray dan peta detail jejak menampilkan node yang terhubung untuk penerbit Amazon SNS, topik Amazon SNS, dan pelanggan hilir:

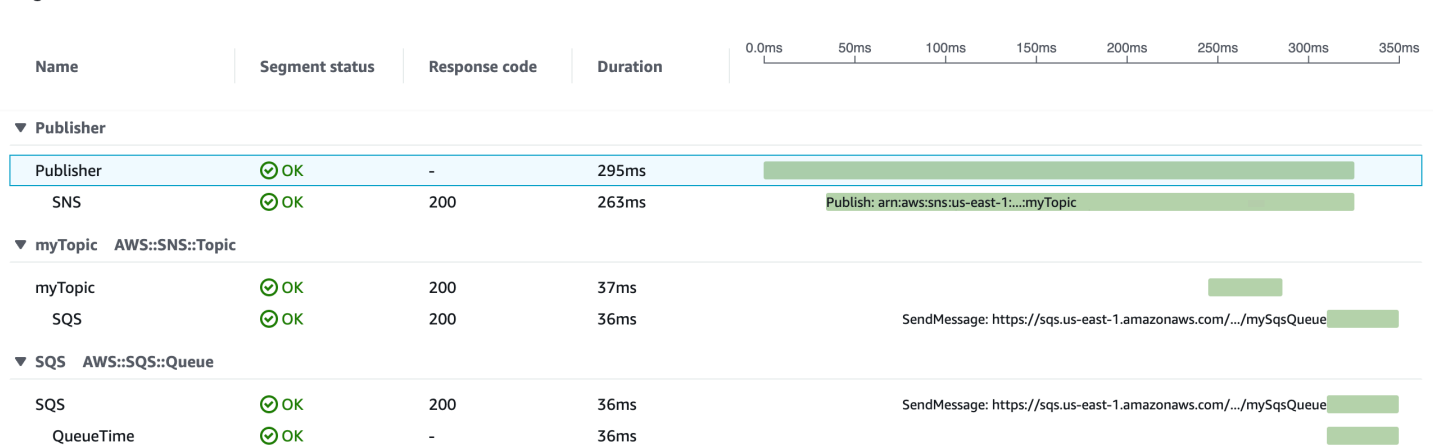


Setelah memilih jejak yang mencakup penerbit dan pelanggan Amazon SNS, halaman detail jejak X-Ray menampilkan peta detail jejak dan garis waktu segmen.

Example Contoh timeline dengan penerbit dan pelanggan Amazon SNS

Contoh ini menunjukkan garis waktu yang menyertakan penerbit Amazon SNS yang mengirim pesan ke topik Amazon SNS, yang diproses oleh pelanggan Amazon SQS.

Segments Timeline [Info](#)

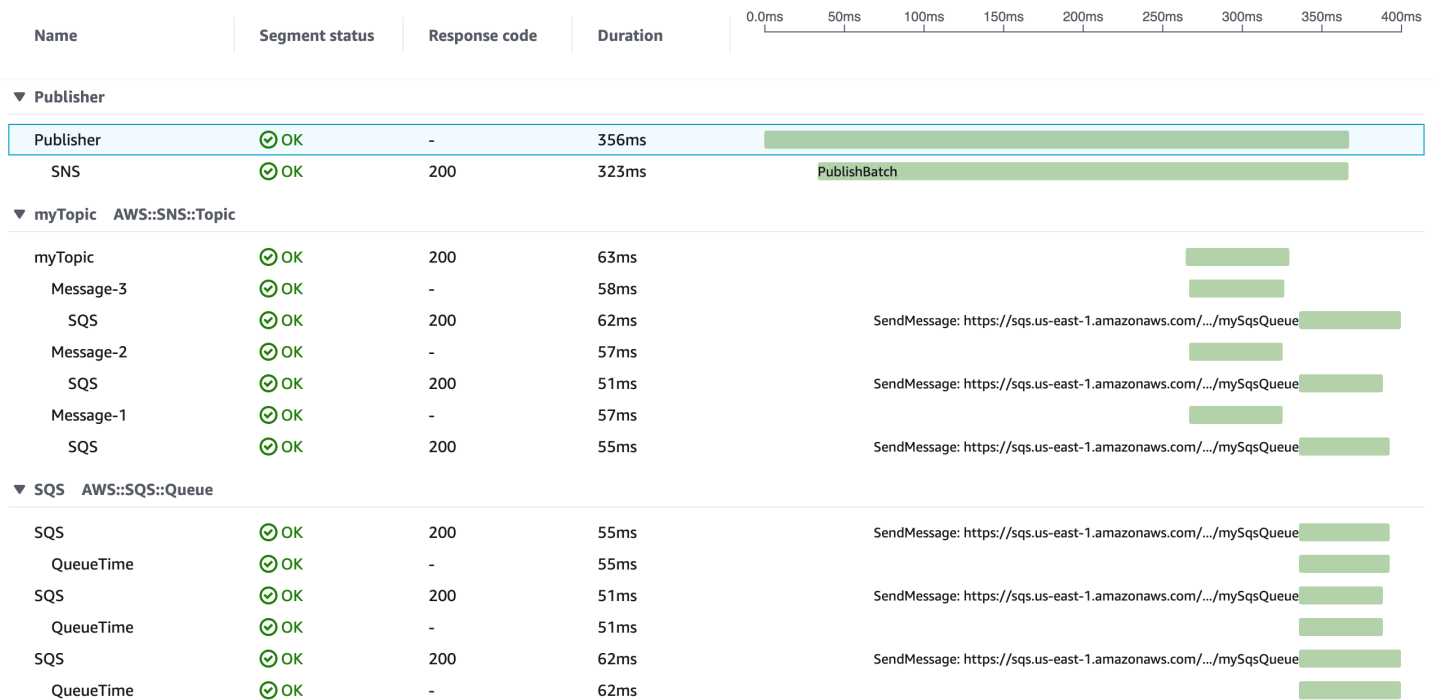


Contoh timeline di atas memberikan detail tentang alur pesan Amazon SNS:

- Segmen SNS mewakili durasi pulang-pergi panggilan Publish API dari klien.
- Segmen myTopic mewakili latensi respons Amazon SNS terhadap permintaan publikasi.
- Subsegmen SQS mewakili waktu pulang-pergi yang dibutuhkan Amazon SNS untuk mempublikasikan pesan ke antrian Amazon SQS.
- Waktu antara segmen MyTopic dan subsegmen SQS mewakili waktu yang dihabiskan pesan dalam sistem Amazon SNS.

Example Contoh timeline dengan pesan Amazon SNS batch

Jika beberapa pesan Amazon SNS dikumpulkan dalam satu jejak, timeline segmen akan menampilkan segmen yang mewakili setiap pesan yang diproses.

Segments Timeline [Info](#)

AWS Step Functions dan AWS X-Ray

AWS X-Ray terintegrasi dengan AWS Step Functions untuk melacak dan menganalisis permintaan untuk Step Functions. Anda dapat memvisualisasikan komponen status mesin, mengidentifikasi hambatan performa, dan memecahkan masalah permintaan yang mengakibatkan kesalahan. Untuk informasi selengkapnya, lihat [AWS X-Ray dan Step Functions](#) di Panduan AWS Step Functions Pengembang.

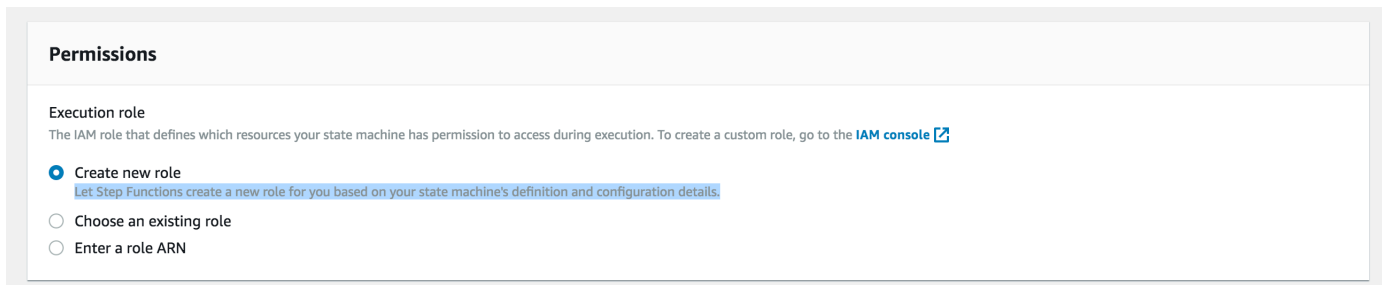
Untuk mengaktifkan penelusuran X-Ray saat membuat mesin status baru

1. Buka konsol Step Functions di <https://console.aws.amazon.com/states/>.
2. Pilih Buat mesin status.
3. Pada halaman Tentukan mesin status, pilih salah satu Penulis dengan potongan kode atau Mulai dengan templat. Jika Anda memilih untuk menjalankan sampel proyek, Anda tidak dapat mengaktifkan Penelusuran X-Ray selama pembuatan. Sebaliknya, mengaktifkan Penelusuran X-Ray setelah Anda membuat mesin keadaan.
4. Pilih Berikutnya.
5. Pada halaman Tentukan Detail, konfigurasi mesin status Anda.

6. Pilih Aktifkan Penelusuran X-Ray.

Cara mengaktifkan penelusuran X-Ray di mesin status yang ada

1. Di konsol Step Functions, pilih mesin status yang Anda ingin mengaktifkan penelusuran.
2. Pilih Edit.
3. Pilih Aktifkan Penelusuran X-Ray.
4. (Opsional) Buat peran baru secara otomatis untuk mesin status Anda untuk menyertakan izin X-Ray dengan memilih Buat peran baru dari jendela Izin.



Permissions

Execution role
The IAM role that defines which resources your state machine has permission to access during execution. To create a custom role, go to the [IAM console](#).

☒ Create new role
[Let Step Functions create a new role for you based on your state machine's definition and configuration details.](#)

☐ Choose an existing role

☐ Enter a role ARN

5. Pilih Simpan.

Note

Saat Anda membuat mesin status baru, mesin ini akan dilacak secara otomatis jika permintaan diambil sampelnya dan penelusuran diaktifkan di layanan upstream seperti Amazon API Gateway atau AWS Lambda. Untuk mesin status yang ada yang tidak dikonfigurasi melalui konsol, misalnya melalui AWS CloudFormation templat, periksa apakah Anda memiliki kebijakan IAM yang memberikan izin yang cukup untuk mengaktifkan jejak X-Ray.

Amazon SQS dan AWS X-Ray

AWS X-Ray terintegrasi dengan Amazon Simple Queue Service (Amazon Simple Queue Service) untuk melacak pesan yang diteruskan melalui antrian Amazon SQS. Jika layanan melacak permintaan dengan menggunakan SDK X-Ray, Amazon SQS dapat mengirim header pelacakan dan terus menyebarkan pelacakan asli dari pengirim ke konsumen dengan ID pelacakan konsisten. Kontinuitas pelacakan mengaktifkan pengguna untuk melacak, menganalisis, dan men-debug di seluruh layanan hilir.

AWS X-Ray mendukung penelusuran aplikasi berbasis peristiwa menggunakan Amazon SQS dan AWS Lambda. Gunakan CloudWatch konsol untuk melihat tampilan tersambung dari setiap permintaan saat diantrian dengan Amazon SQS dan diproses oleh fungsi Lambda hilir. Jejak dari produsen pesan hulu secara otomatis ditautkan ke jejak dari node konsumen Lambda hilir, menciptakan end-to-end tampilan aplikasi. Untuk informasi selengkapnya, lihat [melacak aplikasi berbasis peristiwa](#).



Amazon SQS mendukung instrumentasi header pelacakan berikut:

- Header HTTP default - X-Ray SDK secara otomatis mengisi header jejak sebagai header HTTP saat Anda memanggil Amazon SQS melalui SDK. AWS Header pelacakan default dilakukan oleh `X-Amzn-Trace-Id` dan sesuai dengan semua pesan yang disertakan dalam permintaan [SendMessage](#) atau [SendMessageBatch](#). Untuk mempelajari selengkapnya tentang Header HTTP default, lihat [Header penelusuran](#).
- Atribut Sistem **AWSTraceHeader** – `AWSTraceHeader` adalah [atribut sistem pesan](#) yang disimpan oleh Amazon SQS untuk membawa header pelacakan X-Ray dengan pesan dalam antrian. `AWSTraceHeader` tersedia untuk digunakan meskipun instrumentasi otomatis melalui SDK X-Ray tidak, misalnya saat membuat SDK pelacakan untuk bahasa baru. Ketika kedua instrumentasi header ditetapkan, atribut sistem pesan membatalkan header pelacakan HTTP.

Saat berjalan di Amazon EC2, Amazon SQS mendukung pemrosesan satu pesan pada satu waktu. Ini berlaku saat berjalan di host lokal, dan saat menggunakan layanan kontainer, seperti Amazon ECS, AWS Fargate, atau AWS App Mesh.

Header pelacakan dikecualikan dari ukuran pesan Amazon SQS dan kuota atribut pesan. Mengaktifkan pelacakan X-Ray tidak akan melebihi kuota Amazon SQS Anda. Untuk mempelajari AWS kuota selengkapnya, lihat Kuota [Amazon SQS](#).

Kirim header pelacakan HTTP

Komponen pengirim di Amazon SQS dapat mengirim header pelacakan secara otomatis melalui panggilan [SendMessageBatch](#) atau [SendMessage](#). Ketika klien AWS SDK diinstrumentasi, mereka dapat dilacak secara otomatis melalui semua bahasa yang didukung melalui X-Ray SDK. Ditelusuri Layanan AWS dan sumber daya yang Anda akses dalam layanan tersebut (misalnya, bucket Amazon S3 atau antrean Amazon SQS), muncul sebagai node hilir pada peta jejak di konsol X-Ray.

Untuk mempelajari cara melacak panggilan AWS SDK dengan bahasa pilihan Anda, lihat topik berikut di bagian yang didukung SDKs:

- Go – [Menelusuri panggilan AWS SDK dengan X-Ray SDK for Go](#)
- Java – [Menelusuri panggilan AWS SDK dengan X-Ray SDK for Java](#)
- Node.js – [Menelusuri panggilan AWS SDK dengan X-Ray SDK untuk Node.js](#)
- Python – [Menelusuri panggilan AWS SDK dengan X-Ray SDK untuk Python](#)
- Ruby – [Menelusuri panggilan AWS SDK dengan X-Ray SDK for Ruby](#)
- .NET – [Menelusuri panggilan AWS SDK dengan X-Ray SDK untuk .NET](#)

Mengambil header pelacakan dan memulihkan konteks pelacakan

Jika Anda menggunakan konsumen hilir Lambda, propagasi konteks jejak dilakukan secara otomatis. Untuk melanjutkan propagasi konteks dengan konsumen Amazon SQS lainnya, Anda harus secara manual mengarahkan handoff ke komponen penerima.

Ada tiga langkah utama untuk memulihkan konteks pelacakan:

- Terima pesan dari antrean untuk atribut `AWSTraceHeader` dengan memanggil API [ReceiveMessage](#).
- Mengambil header pelacakan dari atribut.
- Memulihkan ID pelacakan dari header. Secara opsional, tambahkan lebih banyak metrik ke segmen.

Berikut ini adalah contoh implementasi yang ditulis dengan SDK for Java X-Ray.

Example : Mengambil header pelacakan dan memulihkan konteks pelacakan

```
// 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 dan AWS X-Ray

AWS X-Ray terintegrasi dengan Amazon S3 untuk melacak permintaan upstream untuk memperbarui bucket S3 aplikasi Anda. Jika layanan melacak permintaan dengan menggunakan X-Ray SDK, Amazon S3 dapat mengirim header penelusuran ke pelanggan acara hilir seperti, AWS Lambda Amazon SQS, dan Amazon SNS. X-Ray mengaktifkan pesan penelusuran untuk notifikasi peristiwa Amazon S3.

Anda dapat menggunakan peta jejak X-Ray untuk melihat koneksi antara Amazon S3 dan layanan lain yang digunakan aplikasi Anda. Anda juga dapat menggunakan konsol tersebut untuk melihat metrik seperti tingkat latensi dan kegagalan rata-rata. Untuk informasi selengkapnya tentang konsol X-Ray, lihat [Gunakan konsol X-Ray](#).

Amazon S3 mendukung instrumentasi header http default. X-Ray SDK secara otomatis mengisi header jejak sebagai header HTTP saat Anda memanggil Amazon S3 melalui SDK. AWS Header penelusuran default dilakukan oleh X-Amzn-Trace-Id. Untuk mempelajari selengkapnya tentang penelusuran header, lihat [Header penelusuran](#) pada halaman konsep. Propagasi konteks jejak Amazon S3 mendukung pelanggan berikut: Lambda, SQS, dan SNS. Karena SQS dan SNS tidak

memancarkan data segmen itu sendiri, mereka tidak akan muncul di peta jejak atau jejak Anda saat dipicu oleh S3, meskipun mereka akan menyebarkan header penelusuran ke layanan hilir.

Konfigurasi notifikasi Peristiwa Amazon S3

Dengan fitur notifikasi Amazon S3, Anda menerima notifikasi saat peristiwa tertentu terjadi di bucket Anda. Notifikasi ini kemudian dapat disebarkan ke tujuan berikut dalam aplikasi Anda:

- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- AWS Lambda

Untuk daftar peristiwa yang didukung, lihat [Tipe peristiwa yang didukung dalam panduan developer Amazon S3](#).

Amazon SNS dan Amazon SQS

Untuk memublikasikan notifikasi ke topik SNS atau antrean SQS, Anda harus memberikan izin Amazon S3. Untuk memberikan izin ini, Anda melampirkan kebijakan AWS Identity and Access Management (IAM) ke topik SNS tujuan atau antrean SQS. Untuk mempelajari selengkapnya tentang kebijakan IAM yang diperlukan, lihat [Memberikan izin untuk memublikasikan pesan ke topik SNS atau antrean SQS](#).

Untuk informasi tentang mengintegrasikan SNS dan SQS dengan X-Ray lihat, [Amazon SNS dan AWS X-Ray](#) dan [Amazon SQS dan AWS X-Ray](#).

AWS Lambda

Ketika Anda menggunakan konsol Amazon S3 untuk mengonfigurasi notifikasi peristiwa pada bucket S3 untuk fungsi Lambda, konsol tersebut menyiapkan izin yang diperlukan pada fungsi Lambda sehingga Amazon S3 memiliki izin untuk memanggil fungsi dari bucket. Untuk informasi selengkapnya, lihat [Bagaimana Cara Saya Mengaktifkan dan Mengonfigurasi Notifikasi Peristiwa untuk Bucket S3?](#) dalam Panduan Pengguna Konsol Amazon Simple Storage Service.

Anda juga dapat memberikan izin Amazon S3 dari AWS Lambda untuk menjalankan fungsi Lambda Anda. Untuk informasi selengkapnya, lihat [Tutorial: Menggunakan AWS Lambda dengan Amazon S3](#) di Panduan Pengembang AWS Lambda.

Untuk informasi selengkapnya tentang mengintegrasikan Lambda dengan X-Ray, [lihat Menginstrumentasi kode](#) Java di Lambda. AWS

Instrumentasi aplikasi Anda untuk AWS X-Ray

Instrumentasi aplikasi Anda melibatkan pengiriman data jejak untuk permintaan masuk dan keluar dan peristiwa lain dalam aplikasi Anda, bersama dengan metadata tentang setiap permintaan. Ada beberapa opsi instrumentasi berbeda yang dapat Anda pilih atau gabungkan, berdasarkan kebutuhan khusus Anda:

- Instrumentasi otomatis — instrumen aplikasi Anda dengan nol perubahan kode, biasanya melalui perubahan konfigurasi, menambahkan agen instrumentasi otomatis, atau mekanisme lainnya.
- Instrumentasi pustaka — buat perubahan kode aplikasi minimal untuk menambahkan instrumentasi bawaan yang menargetkan pustaka atau kerangka kerja tertentu, seperti AWS SDK, klien HTTP Apache, atau klien SQL.
- Instrumentasi manual — tambahkan kode instrumentasi ke aplikasi Anda di setiap lokasi tempat Anda ingin mengirim informasi jejak.

Ada beberapa agen SDKs, dan alat yang dapat digunakan untuk instrumen aplikasi Anda untuk penelusuran X-Ray.

Topik

- [Menginstrumentasi aplikasi Anda dengan AWS Distro untuk OpenTelemetry](#)
- [Menginstrumentasi aplikasi Anda dengan AWS X-Ray SDKs](#)
- [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#)

Menginstrumentasi aplikasi Anda dengan AWS Distro untuk OpenTelemetry

AWS Distro for OpenTelemetry (ADOT) adalah AWS distribusi berdasarkan proyek Cloud Native Computing Foundation (CNCF). OpenTelemetry menyediakan satu set open source APIs, perpustakaan, dan agen untuk mengumpulkan jejak dan metrik terdistribusi. Toolkit ini adalah distribusi OpenTelemetry komponen hulu termasuk SDKs, agen instrumentasi otomatis, dan kolektor yang diuji, dioptimalkan, diamankan, dan didukung oleh AWS.

Dengan ADOT, para insinyur dapat menginstruksikan aplikasi mereka sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa solusi AWS pemantauan termasuk Amazon CloudWatch, AWS X-Ray dan Amazon Service. OpenSearch

Menggunakan X-Ray dengan ADOT memerlukan dua komponen: OpenTelemetry SDK diaktifkan untuk digunakan dengan X-Ray, dan AWS Distro untuk OpenTelemetry Kolektor diaktifkan untuk digunakan dengan X-Ray. Untuk informasi selengkapnya tentang menggunakan AWS Distro untuk OpenTelemetry with AWS X-Ray dan lainnya Layanan AWS, lihat [AWS Distro untuk OpenTelemetry Dokumentasi](#).

Untuk informasi selengkapnya tentang dukungan dan penggunaan bahasa, lihat [AWS Observabilitas di GitHub](#).

Note

Anda sekarang dapat menggunakan CloudWatch agen untuk mengumpulkan metrik, log, dan jejak dari EC2 instans Amazon dan server di lokasi. CloudWatch agen versi 1.300025.0 dan yang lebih baru dapat mengumpulkan jejak dari atau klien [OpenTelemetry](#) X-Ray SDKs, dan mengirimkannya ke [X-Ray](#). Menggunakan CloudWatch agen alih-alih AWS Distro for OpenTelemetry (ADOT) Collector atau daemon X-Ray untuk mengumpulkan jejak dapat membantu Anda mengurangi jumlah agen yang Anda kelola. Lihat topik [CloudWatch agen](#) di Panduan CloudWatch Pengguna untuk informasi selengkapnya.

ADOT meliputi:

- [AWS Distro untuk Go OpenTelemetry](#)
- [AWS Distro untuk OpenTelemetry Java](#)
- [AWS Distro untuk OpenTelemetry JavaScript](#)
- [AWS Distro untuk Python OpenTelemetry](#)
- [AWS Distro para OpenTelemetry .NET](#)

[ADOT saat ini menyertakan dukungan instrumentasi otomatis untuk Java dan Python. Selain itu, ADOT memungkinkan instrumentasi otomatis fungsi AWS Lambda dan permintaan hilirnya menggunakan runtime Java, Node.js, dan Python, melalui Lapisan Lambda Terkelola ADOT.](#)

ADOT SDKs untuk Java dan Go mendukung aturan pengambilan sampel terpusat X-Ray. Jika Anda memerlukan dukungan untuk aturan sampling X-Ray dalam bahasa lain, pertimbangkan untuk menggunakan AWS X-Ray SDK.

 Note

Anda dapat mengirim sekarang mengirim jejak W3C ke IDs X-Ray. Secara default, jejak yang dibuat dengan OpenTelemetry memiliki format ID jejak yang didasarkan pada spesifikasi [W3C Trace Context](#). Ini berbeda dari format untuk jejak IDs yang dibuat menggunakan X-Ray SDK atau oleh AWS layanan yang terintegrasi dengan X-Ray. Untuk memastikan bahwa jejak IDs dalam format W3C diterima oleh X-Ray, Anda harus menggunakan [AWS X-Ray Exporter](#) versi 0.86.0 atau yang lebih baru, yang disertakan dengan [ADOT](#) Collector versi 0.34.0 dan yang lebih baru. Versi eksportir sebelumnya memvalidasi cap waktu ID jejak, yang dapat menyebabkan jejak W3C ditolak. IDs

Menginstrumentasi aplikasi Anda dengan AWS X-Ray SDKs

AWS X-Ray termasuk satu set bahasa khusus SDKs untuk instrumentasi aplikasi Anda untuk mengirim jejak ke X-Ray. Setiap X-Ray SDK menyediakan yang berikut:

- Interceptors untuk ditambahkan ke kode Anda untuk melacak permintaan HTTP yang masuk
- Penangan klien untuk instrumen klien AWS SDK yang digunakan aplikasi Anda untuk memanggil orang lain Layanan AWS
- Klien HTTP untuk panggilan instrumen ke layanan web HTTP internal dan eksternal lainnya

X-Ray SDKs juga mendukung panggilan instrumentasi ke database SQL, instrumentasi klien AWS SDK otomatis, dan fitur lainnya. Daripada mengirim data pelacakan langsung ke X-Ray, SDK mengirim dokumen segmen JSON ke proses daemon yang mendengarkan lalu lintas UDP. [X-Ray Daemon](#) menyangga segmen dalam antrean dan mengunggah segmen tersebut ke X-Ray dalam batch.

Khusus bahasa berikut disediakan SDKs :

- [AWS X-Ray SDK for Go](#)
- [AWS X-Ray SDK for Java](#)
- [AWS X-Ray SDK untuk Node.js](#)
- [AWS X-Ray SDK untuk Python](#)
- [AWS X-Ray SDK for .NET](#)
- [AWS X-Ray SDK for Ruby](#)

[X-Ray saat ini menyertakan dukungan instrumentasi otomatis untuk Java.](#)

Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs

Yang SDKs disertakan dengan X-Ray adalah bagian dari solusi instrumentasi terintegrasi yang ditawarkan oleh AWS. AWS Distro for OpenTelemetry adalah bagian dari solusi industri yang lebih luas di mana X-Ray hanyalah salah satu dari banyak solusi penelusuran. Anda dapat menerapkan end-to-end penelusuran di X-Ray menggunakan salah satu pendekatan, tetapi penting untuk memahami perbedaan untuk menentukan pendekatan yang paling berguna bagi Anda.

Kami merekomendasikan instrumentasi aplikasi Anda dengan AWS Distro untuk OpenTelemetry jika Anda membutuhkan yang berikut:

- Kemampuan untuk mengirim jejak ke beberapa ujung belakang penelusuran yang berbeda tanpa harus menginstruksikan ulang kode Anda
- Support untuk sejumlah besar instrumentasi perpustakaan untuk setiap bahasa, dikelola oleh komunitas OpenTelemetry
- Lapisan Lambda yang dikelola sepenuhnya yang mengemas semua yang Anda butuhkan untuk mengumpulkan data telemetri, tanpa memerlukan perubahan kode saat menggunakan Java, Python, atau Node.js

Note

AWS Distro for OpenTelemetry menawarkan pengalaman memulai yang lebih sederhana untuk menginstrumentasi fungsi Lambda Anda. Namun, karena OpenTelemetry penawaran fleksibilitas, fungsi Lambda Anda akan memerlukan memori tambahan dan pemanggilan mungkin mengalami peningkatan latensi start dingin, yang dapat menyebabkan biaya tambahan. Jika Anda mengoptimalkan latensi rendah dan tidak memerlukan OpenTelemetry kemampuan lanjutan seperti tujuan back end yang dapat dikonfigurasi secara dinamis, Anda mungkin ingin menggunakan AWS X-Ray SDK untuk instrumen aplikasi Anda.

Sebaiknya pilih X-Ray SDK untuk menginstrumentasi aplikasi Anda jika Anda memerlukan yang berikut:

- Solusi vendor tunggal yang terintegrasi erat

- Integrasi dengan aturan sampling terpusat X-Ray, termasuk kemampuan untuk mengonfigurasi aturan pengambilan sampel dari konsol X-Ray dan secara otomatis menggunakannya di beberapa host, saat menggunakan Node.js, Python, Ruby, atau .NET

Pencarian Transaksi

Transaction Search adalah pengalaman analitik interaktif yang dapat Anda gunakan untuk mendapatkan visibilitas lengkap dari rentang transaksi aplikasi Anda. Rentang adalah unit dasar operasi dalam jejak terdistribusi dan mewakili tindakan atau tugas tertentu dalam aplikasi atau sistem. Setiap rentang mencatat rincian tentang segmen tertentu dari transaksi. Detail ini mencakup waktu mulai dan akhir, durasi, dan metadata terkait, yang dapat mencakup atribut bisnis seperti pelanggan IDs dan pesanan. IDs Rentang diatur dalam hierarki orang tua-anak. Hierarki ini membentuk jejak lengkap yang memetakan aliran transaksi di berbagai komponen atau layanan.

Untuk informasi selengkapnya, lihat [Pencarian Transaksi](#).

OpenTelemetry Titik Akhir Protokol (OTLP)

OpenTelemetry adalah kerangka kerja observabilitas sumber terbuka yang menyediakan tim TI dengan protokol dan alat standar untuk mengumpulkan dan merutekan data telemetri. Ini memberikan format terpadu untuk menginstrumentasi, menghasilkan, mengumpulkan, dan mengeksport data telemetri aplikasi, seperti metrik, log, dan jejak ke platform pemantauan untuk analisis dan wawasan. Dengan menggunakan OpenTelemetry, tim dapat menghindari penguncian vendor, memastikan fleksibilitas dalam solusi observabilitas mereka.

[Anda dapat menggunakan OpenTelemetry untuk langsung mengirim jejak ke titik akhir OpenTelemetry Protokol \(OTLP\), dan mendapatkan pengalaman pemantauan kinerja aplikasi out-of-the kotak di CloudWatch Sinyal Aplikasi.](#)

Untuk informasi selengkapnya, lihat [OpenTelemetry](#).

Bekerja dengan Go

Ada dua cara untuk instrumen aplikasi Go Anda untuk mengirim jejak ke X-Ray:

- [AWS Distro for OpenTelemetry Go](#) — AWS Distribusi yang menyediakan serangkaian pustaka open source untuk mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan termasuk Amazon CloudWatch,, AWS X-Ray dan Amazon OpenSearch Service, melalui Distro for Collector.AWS OpenTelemetry
- [AWS X-Ray SDK for Go](#) — Satu set perpustakaan untuk menghasilkan dan mengirim jejak ke X-Ray melalui daemon X-Ray.

Untuk informasi selengkapnya, lihat [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#).

AWS Distro untuk Go OpenTelemetry

Dengan AWS Distro for OpenTelemetry Go, Anda dapat menginstruksikan aplikasi Anda sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa solusi AWS pemantauan termasuk Amazon CloudWatch,, AWS X-Ray dan Amazon Service. OpenSearch Menggunakan X-Ray dengan AWS Distro untuk OpenTelemetry membutuhkan dua komponen: OpenTelemetry SDK diaktifkan untuk digunakan dengan X-Ray, dan AWS Distro untuk OpenTelemetry Kolektor diaktifkan untuk digunakan dengan X-Ray.

Untuk memulai, lihat [dokumentasi AWS Distro for OpenTelemetry Go](#).

Untuk informasi selengkapnya tentang menggunakan AWS Distro untuk OpenTelemetry with AWS X-Ray dan lainnya Layanan AWS, lihat [AWS Distro untuk OpenTelemetry atau Distro untuk AWS Dokumentasi](#). OpenTelemetry

Untuk informasi selengkapnya tentang dukungan dan penggunaan bahasa, lihat [AWS Observabilitas di GitHub](#).

AWS X-Ray SDK for Go

X-Ray SDK for Go adalah serangkaian pustaka untuk aplikasi Go yang menyediakan kelas dan metode untuk menghasilkan dan mengirim pelacakan data ke daemon X-Ray. Data pelacakan mencakup informasi tentang permintaan HTTP masuk yang disajikan oleh aplikasi, dan panggilan

yang dilakukan aplikasi ke layanan hilir menggunakan AWS SDK, klien HTTP, atau konektor database SQL. Anda juga dapat membuat segmen secara manual dan menambahkan informasi debug dalam anotasi dan metadata.

Unduh SDK dari [GitHub repositori](https://github.com/aws/aws-xray-sdk-go) dengan `go get`:

```
$ go get -u github.com/aws/aws-xray-sdk-go/...
```

Untuk aplikasi web, mulai dengan [menggunakan `xray.Handler` fungsi](#) untuk melacak permintaan yang masuk. Pesan handler membuat [segmen](#) untuk setiap permintaan yang dilacak, dan melengkapi segmen ketika respons dikirim. Ketika segmen terbuka Anda dapat menggunakan metode klien SDK untuk menambahkan informasi ke segmen dan membuat subsegmen untuk pelacakan panggilan hilir. SDK juga secara otomatis mencatat pengecualian yang aplikasi Anda lempar ketika segmen terbuka.

Untuk fungsi Lambda disebut oleh instrumen aplikasi atau layanan, Lambda membaca [tracing header](#) dan pelacakan sampel permintaan secara otomatis. Untuk fungsi lainnya, Anda dapat [mengonfigurasi Lambda](#) untuk sampel dan pelacakan permintaan masuk. Dalam kedua kasus, Lambda membuat segmen dan menyediakannya ke X-Ray SDK.

Note

Pada Lambda, X-Ray SDK adalah opsional. Jika Anda tidak menggunakannya dalam fungsi Anda, peta layanan Anda masih akan menyertakan simpul untuk layanan Lambda, dan satu untuk setiap fungsi Lambda. Dengan menambahkan SDK, Anda dapat melakukan instrumen kode fungsi Anda untuk menambahkan subsegmen ke segmen fungsi yang dicatat oleh Lambda. Lihat [AWS Lambda dan AWS X-Ray](#) untuk informasi selengkapnya.

Selanjutnya, [bungkus klien Anda dengan panggilan ke AWS fungsi](#). Langkah ini memastikan bahwa instrumen X-Ray memanggil metode klien apa pun. Anda juga dapat [instrumen panggilan ke basis data SQL](#).

Setelah Anda mulai menggunakan SDK, sesuaikan perilakunya dengan [mengonfigurasi perekam dan middleware](#). Anda dapat menambahkan plugin untuk mencatat data mengenai sumber daya komputasi yang berjalan di aplikasi Anda, menyesuaikan perilaku sampling dengan mendefinisikan aturan sampling, dan mengatur tingkat log untuk melihat lebih atau kurang informasi dari SDK dalam log aplikasi Anda.

Catat informasi tambahan tentang permintaan dan pekerjaan yang dilakukan aplikasi Anda dalam [anotasi dan metadata](#). Anotasi adalah pasangan kunci-nilai sederhana yang diindeks untuk digunakan dengan [ekspresi filter](#), sehingga Anda dapat mencari pelacakan yang berisi data tertentu. Entri metadata kurang bersifat membatasi dan dapat mencatat seluruh objek dan array — segala yang dapat disambungkan ke dalam JSON.

Anotasi dan Metadata

Anotasi dan metadata adalah teks abritari yang Anda tambahkan ke segmen dengan X-Ray SDK. Anotasi diindekskan untuk digunakan dengan ekspresi filter. Metadata tidak diindeks, tetapi dapat dilihat di segmen mentah dengan konsol X-Ray atau API. Siapa pun yang Anda berikan akses baca ke X-Ray dapat melihat data ini.

Bila Anda memiliki banyak klien diinstrumentasi dalam kode Anda, segmen permintaan tunggal dapat berisi sejumlah besar subsegmen, satu untuk setiap panggilan yang dilakukan dengan klien yang diinstrumentasi. Anda dapat mengatur dan mengelompokkan subsegmen dengan menggabungkan panggilan klien di [subsegmen kustom](#). Anda dapat membuat subsegmen kustom untuk seluruh fungsi atau bagian dari kode apa pun, dan mencatat metadata dan anotasi pada subsegmen alih-alih menulis semuanya pada segmen induk.

Persyaratan

X-Ray SDK for Go memerlukan Go 1.9 atau versi yang lebih baru.

SDK tergantung pada pustaka berikut saat kompilasi dan waktu aktif:

- AWS SDK for Go versi 1.10.0 atau yang lebih baru

Ketergantungan ini dinyatakan dalam file README.md.

Referensi dokumentasi

Setelah Anda mengunduh SDK, membangun dan meng-host dokumentasi lokal untuk melihatnya di web peramban.

Untuk melihat referensi dokumentasi

1. Navigasikan ke direktori `$GOPATH/src/github.com/aws/aws-xray-sdk-go` (Linux atau Mac) atau folder `%GOPATH%\src\github.com\aws\aws-xray-sdk-go` (Windows)
2. Jalankan perintah `godoc`.

```
$ godoc -http=:6060
```

3. Membuka peramban di `http://localhost:6060/pkg/github.com/aws/aws-xray-sdk-go/`.

Mengonfigurasi X-Ray SDK for Go

Anda dapat menentukan konfigurasi untuk X-Ray SDK for Go through variabel lingkungan, dengan `Configure` memanggil dengan objek, atau `Config` dengan mengasumsikan nilai default. Variabel lingkungan lebih diutamakan dibandingkan nilai `Config`, yang lebih diutamakan dibandingkan nilai default apa pun.

Bagian-bagian

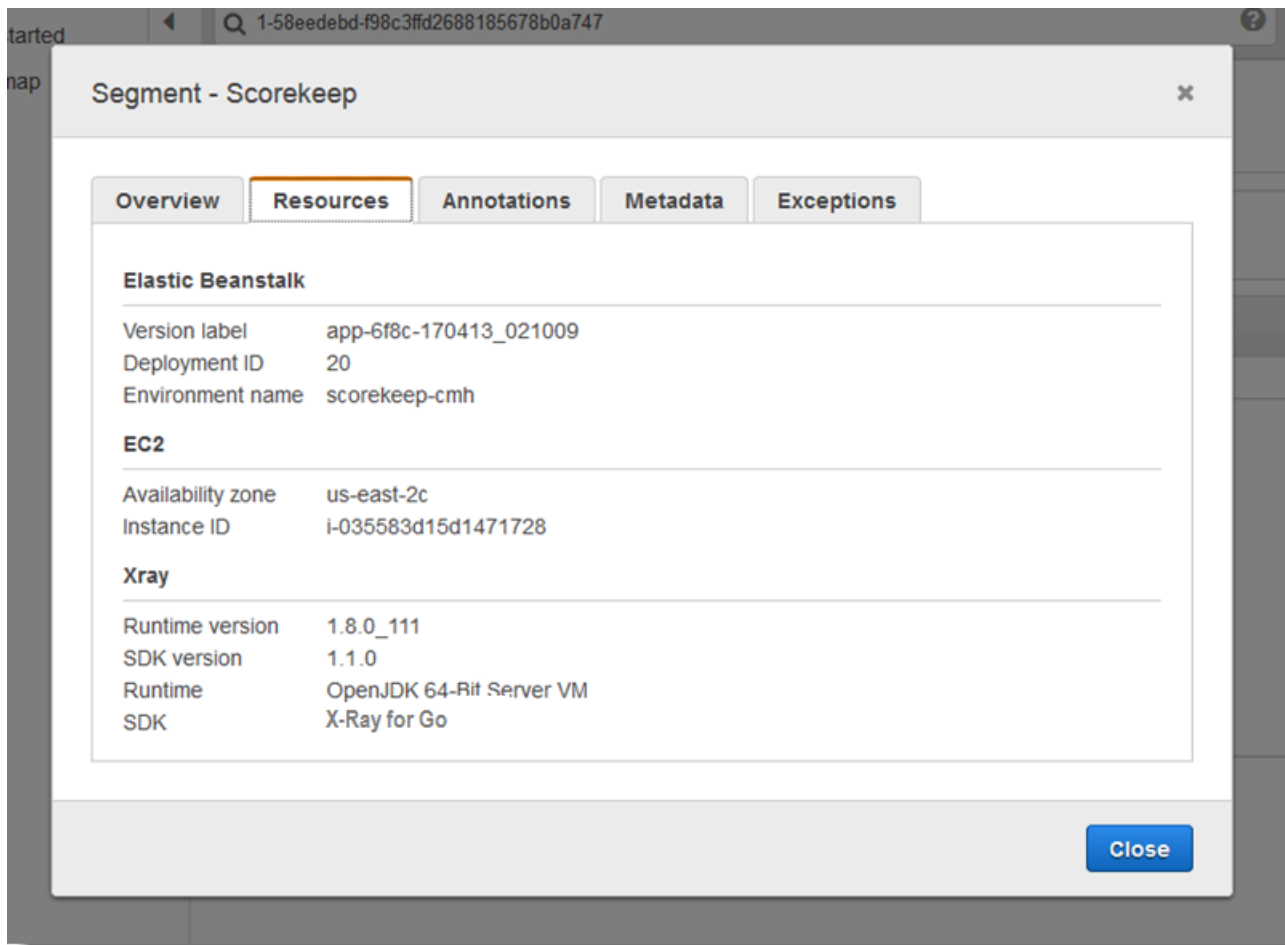
- [Plugin layanan](#)
- [Aturan pengambilan sampel](#)
- [Pencatatan log](#)
- [Variabel-variabel lingkungan](#)
- [Menggunakan konfigurasi](#)

Plugin layanan

Gunakan `plugins` untuk mencatat informasi tentang layanan yang meng-hosting aplikasi Anda.

Plugin

- Amazon EC2 — `EC2Plugin` menambahkan ID instans, Availability Zone, dan Grup CloudWatch Log.
- Elastic Beanstalk – `ElasticBeanstalkPlugin` menambahkan nama lingkungan, label versi, dan ID deployment.
- Amazon ECS – `ECSPlugin` menambahkan ID kontainer.



Untuk menggunakan plugin, impor salah satu paket berikut.

```
"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"
```

Setiap plugin memiliki panggilan fungsi `Init()` eksplisit yang memuat 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",
})
}
```

SDK juga menggunakan pengaturan plugin untuk mengatur bidang origin di segmen. Ini menunjukkan jenis AWS sumber daya yang menjalankan aplikasi Anda. Saat Anda menggunakan beberapa plugin, SDK menggunakan urutan resolusi berikut untuk menentukan asal: ElasticBeanstalk > EKS > ECS > EC2

Aturan pengambilan sampel

SDK menggunakan aturan pengambilan sampel yang Anda tetapkan di konsol X-Ray untuk menentukan permintaan yang akan dicatat. Aturan default menelusuri permintaan pertama setiap detik, dan lima persen permintaan tambahan di semua layanan yang mengirim pelacakan ke X-Ray. [Buat aturan tambahan di konsol X-Ray](#) untuk menyesuaikan jumlah data yang dicatat untuk setiap aplikasi Anda.

SDK menerapkan aturan kustom sesuai urutan penetapannya. Jika permintaan cocok dengan beberapa aturan kustom, SDK hanya menerapkan aturan pertama.

Note

Jika SDK tidak dapat mencapai X-Ray untuk mendapatkan aturan pengambilan sampel, SDK akan beralih ke aturan lokal default dari permintaan pertama setiap detik, dan lima persen permintaan tambahan per host. Hal ini dapat terjadi jika host tidak memiliki izin untuk memanggil sampling APIs, atau tidak dapat terhubung ke daemon X-Ray, yang bertindak sebagai proxy TCP untuk panggilan API yang dilakukan oleh SDK.

Anda juga dapat mengonfigurasi SDK untuk memuat aturan sampling dari dokumen JSON. SDK dapat menggunakan aturan lokal sebagai cadangan jika terjadi kasus tidak dapat mengambil sampel X-Ray, atau menggunakan aturan lokal secara eksklusif.

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
  }
}
```

Contoh ini menentukan satu aturan kustom dan aturan default. Aturan kustom menerapkan tingkat pengambilan sampel lima persen tanpa jumlah minimum permintaan untuk melacak jalur di `/api/move/`. Aturan default menelusuri permintaan pertama setiap detik dan 10 persen dari permintaan tambahan.

Kerugian dari menentukan aturan secara lokal adalah bahwa target tetap diterapkan oleh setiap instans pencatat secara independen, alih-alih dikelola oleh layanan X-Ray. Ketika Anda men-deploy lebih banyak host, laju tetap akan dikalikan, sehingga sulit untuk mengontrol jumlah data yang dicatat.

Pada AWS Lambda, Anda tidak dapat mengubah laju pengambilan sampel. Jika fungsi Anda dipanggil oleh layanan yang diinstrumentasikan, panggilan yang menghasilkan permintaan yang sampelnya diambil oleh layanan yang akan dicatat oleh Lambda. Jika pelacakan aktif diaktifkan dan tidak ada header pelacakan, Lambda membuat keputusan pengambilan sampel.

Untuk memberikan aturan pencadangan, arahkan ke file JSON pengambilan sampel lokal dengan menggunakan `NewCentralizedStrategyWithFilePath`.

Example main.go - Aturan pengambilan sampel lokal

```
s, _ := sampling.NewCentralizedStrategyWithFilePath("sampling.json") // path to local
sampling json
```

```
xray.Configure(xray.Config{SamplingStrategy: s})
```

Untuk hanya menggunakan aturan lokal, arahkan ke file JSON pengambilan sampel lokal dengan menggunakan `NewLocalizedStrategyFromFilePath`.

Example main.go – Nonaktifkan pengambilan sampel

```
s, _ := sampling.NewLocalizedStrategyFromFilePath("sampling.json") // path to local
sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

Pencatatan log

Note

Bidang `xray.Config{}` `LogLevel` dan `LogFormat` tidak digunakan lagi mulai dari versi 1.0.0-rc.10.

X-Ray menggunakan antarmuka berikut untuk pencatatan. Pencatat default menulis ke `stdout` di `LogLevelInfo` dan di atasnya.

```
type Logger interface {
    Log(level LogLevel, msg fmt.Stringer)
}

const (
    LogLevelDebug LogLevel = iota + 1
    LogLevelInfo
    LogLevelWarn
    LogLevelError
)
```

Example tulis ke **io.Writer**

```
xray.SetLogger(xraylog.NewDefaultLogger(os.Stderr, xraylog.LogLevelError))
```

Variabel-variabel lingkungan

Anda dapat menggunakan variabel lingkungan untuk mengonfigurasi X-Ray SDK for Go. SDK mendukung variabel berikut.

- `AWS_XRAY_CONTEXT_MISSING`— Setel `RUNTIME_ERROR` untuk melempar pengecualian saat kode instrumentasi Anda mencoba merekam data saat tidak ada segmen yang terbuka.

Nilai Valid

- `RUNTIME_ERROR`— Lempar pengecualian runtime.
- `LOG_ERROR`— Log kesalahan dan lanjutkan (default).
- `IGNORE_ERROR`— Abaikan kesalahan dan lanjutkan.

Kesalahan yang berkaitan dengan segmen atau subsegmen yang hilang dapat terjadi ketika Anda mencoba untuk menggunakan klien yang diinstrumentasi dalam kode perusahaan rintisan yang berjalan ketika tidak ada permintaan terbuka, atau dalam kode yang memunculkan thread baru.

- `AWS_XRAY_TRACING_NAME` – Tetapkan nama layanan yang digunakan SDK untuk segmen.
- `AWS_XRAY_DAEMON_ADDRESS` – Atur host dan port listener daemon X-Ray. Secara default, SDK mengirimkan pelacakan data ke `127.0.0.1:2000`. Gunakan variabel ini jika Anda telah mengonfigurasi daemon untuk [mendengarkan pada port yang berbeda](#) atau jika dijalankan pada host yang berbeda.
- `AWS_XRAY_CONTEXT_MISSING` – Atur nilai untuk menentukan cara SDK menangani kesalahan konteks yang hilang. Kesalahan yang terkait dengan segmen atau subsegmen yang hilang dapat terjadi saat Anda mencoba menggunakan klien yang diinstrumentasikan dalam kode startup saat tidak ada permintaan yang terbuka, atau dalam kode yang memunculkan utas baru.
 - `RUNTIME_ERROR` – Secara default, SDK diatur untuk mengeluarkan pengecualian waktu aktif.
 - `LOG_ERROR` – Atur untuk mencatat kesalahan dan melanjutkan.

Variabel lingkungan mengganti nilai yang setara yang diatur dalam kode.

Menggunakan konfigurasi

Anda juga dapat mengonfigurasi X-Ray SDK for Go menggunakan metode `Configure`. `Configure` mengambil satu argumen, sebuah objek `Config`, dengan bidang opsional berikut.

DaemonAddr

String ini menentukan host dan port dari listener daemon X-Ray. Jika tidak ditentukan, X-Ray menggunakan nilai variabel lingkungan `AWS_XRAY_DAEMON_ADDRESS`. Jika nilai tersebut tidak diatur, X-Ray menggunakan "127.0.0.1:2000".

ServiceVersion

String ini menentukan versi layanan. Jika tidak ditentukan, X-Ray menggunakan string kosong (`""`).

SamplingStrategy

Objek `SamplingStrategy` ini menentukan panggilan aplikasi Anda yang dilacak. Jika tidak ditentukan, X-Ray menggunakan `LocalizedSamplingStrategy`, yang mengambil strategi seperti yang ditentukan dalam `xray/resources/DefaultSamplingRules.json`.

StreamingStrategy

`StreamingStrategy` Objek ini menentukan apakah untuk streaming segmen ketika `RequiresStreaming` mengembalikan `true`. Jika tidak ditentukan, X-Ray menggunakan `DefaultStreamingStrategy` yang mengalirkan segmen sampel jika jumlah subsegmen lebih besar dari 20.

ExceptionFormattingStrategy

Objek `ExceptionFormattingStrategy` ini menentukan cara yang Anda inginkan dalam menangani berbagai pengecualian. Jika tidak ditentukan, X-Ray menggunakan `DefaultExceptionFormattingStrategy` dengan `XrayError` dari tipe error, pesan kesalahan, dan pelacakan tumpukan.

Menginstrumentasikan permintaan HTTP yang masuk dengan X-Ray SDK for Go

Anda dapat menggunakan X-Ray SDK untuk melacak permintaan HTTP masuk yang dilayani aplikasi Anda pada EC2 instans di Amazon EC2 AWS Elastic Beanstalk, atau Amazon ECS.

Gunakan `xray.Handler` untuk menginstrumentasikan permintaan HTTP yang masuk. X-Ray SDK for Go menerapkan antarmuka `http.Handler` pustaka Go standar di kelas `xray.Handler` untuk menghalangi permintaan web. Kelas `xray.Handler` membungkus `http.Handler` yang disediakan

dengan `xray.Capture` menggunakan konteks permintaan, mengurai header masuk, menambahkan header respons jika diperlukan, dan menetapkan bidang pelacakan khusus HTTP.

Bila Anda menggunakan kelas ini untuk menangani permintaan dan respons HTTP, X-Ray SDK for Go membuat segmen untuk setiap permintaan sampel. Segmen ini mencakup waktu, metode, dan disposisi permintaan HTTP. Instrumentasi tambahan membuat subsegmen pada segmen ini.

Note

Untuk AWS Lambda fungsi, Lambda membuat segmen untuk setiap permintaan sampel. Untuk informasi selengkapnya, lihat [AWS Lambda dan AWS X-Ray](#).

Contoh berikut menghalangi permintaan pada port 8000 dan mengirim "Halo!" sebagai tanggapan. Hal ini menciptakan segmen `myApp` dan panggilan instrumen melalui aplikasi apa pun.

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)
}
```

Setiap segmen memiliki nama yang mengidentifikasi aplikasi Anda dalam peta layanan. Segmen dapat diberi nama secara statis, atau Anda dapat mengonfigurasi SDK untuk nama itu secara dinamis berdasarkan header host dalam permintaan masuk. Penamaan dinamis memungkinkan Anda mengelompokkan pelacakan berdasarkan nama domain dalam permintaan, dan menerapkan nama default jika nama tersebut tidak cocok dengan pola yang diharapkan (misalnya, jika header host ditiru).

Permintaan yang Diteruskan

Jika penyeimbang beban atau perantara lainnya meneruskan permintaan ke aplikasi Anda, X-Ray akan mengambil IP klien dari header `X-Forwarded-For` dalam permintaan bukan

dari sumber IP dalam paket IP. IP klien yang dicatat untuk permintaan yang diteruskan dapat ditiru, sehingga tidak dapat dipercaya.

Ketika permintaan diteruskan, SDK menetapkan bidang tambahan di segmen untuk menunjukkan ini. Jika segmen yang berisi bidang `x_forwarded_for` ditetapkan ke `true`, IP klien diambil dari header `X-Forwarded-For` dalam permintaan HTTP.

Penangan membuat segmen untuk setiap permintaan masuk dengan blok `http` yang berisi informasi berikut:

- Metode HTTP – DAPATKAN, POSTING, LETAKKAN, HAPUS, dll.
- Alamat klien – Alamat IP klien yang mengirim permintaan.
- Kode respons – Kode respons HTTP untuk permintaan yang selesai.
- Timing – Waktu mulai (saat permintaan diterima) dan waktu akhir (saat respons dikirim).
- Agen pengguna — `user-agent` dari permintaan.
- Panjang konten — `content-length` dari respons.

Mengonfigurasi strategi penamaan segmen

AWS X-Ray menggunakan nama layanan untuk mengidentifikasi aplikasi Anda dan membedakannya dari aplikasi lain, database, eksternal APIs, dan AWS sumber daya yang digunakan aplikasi Anda. Saat SDK X-Ray membuat segmen untuk permintaan masuk, SDK akan mencatat nama layanan aplikasi Anda di [kolom nama](#).

SDK X-Ray dapat memberi nama segmen setelah nama host di header permintaan HTTP. Namun, header ini dapat ditiru, yang dapat mengakibatkan simpul tak terduga di peta layanan Anda. Untuk mencegah SDK dari penamaan segmen salah karena permintaan dengan header host palsu, Anda harus menentukan nama default untuk permintaan masuk.

Jika aplikasi Anda menyuguhkan permintaan untuk beberapa domain, Anda dapat mengonfigurasi SDK untuk menggunakan strategi penamaan dinamis untuk mencerminkan ini dalam nama segmen. Strategi penamaan dinamis mengizinkan SDK menggunakan nama host untuk permintaan yang sesuai dengan pola yang diharapkan, dan menerapkan nama default untuk permintaan yang tidak sesuai.

Misalnya, Anda boleh memiliki satu aplikasi yang melayani permintaan untuk tiga subdomain—`www.example.com`, `api.example.com`, dan `static.example.com`. Anda dapat menggunakan

strategi penamaan dinamis dengan pola `*.example.com` untuk mengidentifikasi segmen untuk setiap subdomain dengan nama yang berbeda, mengakibatkan tiga simpul layanan pada peta layanan. Jika aplikasi Anda menerima permintaan dengan nama host yang tidak cocok dengan pola, Anda akan melihat simpul keempat pada peta layanan dengan nama fallback yang Anda tentukan.

Untuk menggunakan nama yang sama untuk semua segmen permintaan, tentukan nama aplikasi Anda saat membuat penanganan, seperti yang ditampilkan dalam bagian sebelumnya.

Note

Anda dapat menimpa nama layanan default yang Anda tentukan dalam kode dengan [variabel lingkungan](#) `AWS_XRAY_TRACING_NAME`.

Strategi penamaan dinamis menentukan pola yang harus sesuai dengan nama host, dan nama default untuk digunakan jika nama host dalam permintaan HTTP tidak cocok dengan pola. Untuk penamaan segmen secara dinamis, gunakan `NewDynamicSegmentNamer` untuk mengonfigurasi nama default dan pola agar sesuai.

Example `main.go`

Jika nama host dalam permintaan cocok dengan pola `*.example.com`, gunakan nama host. Jika tidak sesuai, gunakan `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)
}
```

Menelusuri panggilan AWS SDK dengan X-Ray SDK for Go

[Saat aplikasi Anda melakukan panggilan Layanan AWS untuk menyimpan data, menulis ke antrian, atau mengirim notifikasi, X-Ray SDK for Go melacak panggilan hilir di subsegmen.](#) Ditelusuri Layanan AWS dan sumber daya yang Anda akses dalam layanan tersebut (misalnya, bucket Amazon S3 atau antrian Amazon SQS), muncul sebagai node hilir pada peta jejak di konsol X-Ray.

Untuk melacak klien AWS SDK, bungkus objek klien dengan panggilan `xray.AWS()` seperti yang ditunjukkan dalam contoh berikut.

Example main.go

```
var dynamo *dynamodb.DynamoDB
func main() {
    dynamo = dynamodb.New(session.Must(session.NewSession()))
    xray.AWS(dynamo.Client)
}
```

Lalu, ketika Anda menggunakan klien AWS SDK, gunakan versi `withContext` dari metode panggilan, dan teruskan `context` dari `http.Request` objek yang diteruskan ke [Handler](#).

Example main.go — AWS Panggilan SDK

```
func listTablesWithContext(ctx context.Context) {
    output := dynamo.ListTablesWithContext(ctx, &dynamodb.ListTablesInput{})
    doSomething(output)
}
```

Untuk semua layanan, Anda dapat melihat nama API yang dipanggil di konsol X-Ray. Untuk subset layanan, X-Ray SDK menambahkan informasi ke segmen untuk memberikan lebih banyak perincian di peta layanan.

Sebagai contoh, ketika Anda melakukan panggilan dengan klien DynamoDB berinstrumen, SDK menambahkan nama tabel ke segmen untuk panggilan yang menargetkan tabel. Di konsol tersebut, setiap tabel muncul sebagai simpul terpisah di peta layanan, dengan simpul DynamoDB generik untuk panggilan yang tidak menargetkan tabel.

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
```

```

    "content_length": 60,
    "status": 200
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}

```

Ketika Anda mengakses sumber daya bernama, panggilan ke layanan berikut membuat simpul tambahan di peta layanan. Panggilan yang tidak menargetkan sumber daya tertentu membuat simpul generik untuk layanan tersebut.

- Amazon DynamoDB – Nama tabel
- Amazon Simple Storage Service – Bucket dan nama kunci
- Amazon Simple Queue Service – Nama antrean

Menelusuri panggilan ke layanan web HTTP downstream dengan X-Ray SDK for Go

Ketika aplikasi Anda membuat panggilan ke microservices atau HTTP publik APIs, Anda dapat menggunakan `xray.Client` untuk instrumen panggilan tersebut sebagai subsegmen dari aplikasi Go Anda, seperti yang ditunjukkan dalam contoh berikut, di mana `http-client` adalah klien HTTP.

Klien membuat salinan dangkal dari klien HTTP yang disediakan, default untuk `http.DefaultClient`, dengan `roundtripper` yang dibungkus dengan `xray.RoundTripper`.

Example

<caption>main.go – klien HTTP</caption>

```
myClient := xray.Client(http-client)
```

<caption>main.go - Telusuri panggilan HTTP downstream dengan pustaka `ctxhttp`</caption>

Contoh berikut instrumen panggilan HTTP keluar dengan pustaka `ctxhttp` menggunakan `xray.Client`. `ctx` dapat diteruskan dari panggilan upstream. Hal ini memastikan bahwa konteks

segmen yang ada digunakan. Misalnya, X-Ray tidak mengizinkan segmen baru dibuat dalam fungsi Lambda, sehingga konteks segmen Lambda yang ada harus digunakan.

```
resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

Menelusuri Querri SQL dengan X-Ray SDK for Go

Untuk menelusuri panggilan SQL ke PostgreSQL atau MySQL, mengganti panggilan `sql.Open` ke `xray.SQLContext`, seperti yang ditunjukkan dalam contoh berikut. Gunakan URLs alih-alih string konfigurasi jika memungkinkan.

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
}
```

Membuat subsegmen kustom dengan X-Ray SDK for Go

Subsegmen memperpanjang [segmen](#) penelusuran dengan detail tentang pekerjaan yang dilakukan untuk melayani permintaan. Setiap kali Anda melakukan panggilan dengan klien berinstrumen, X-Ray tersebut mencatat informasi yang dihasilkan dalam subsegmen. Anda dapat membuat subsegmen tambahan untuk mengelompokkan subsegmen lain, untuk mengukur performa bagian kode, atau untuk mencatat anotasi dan metadata.

Gunakan metode `Capture` untuk membuat subsegmen sekitar fungsi.

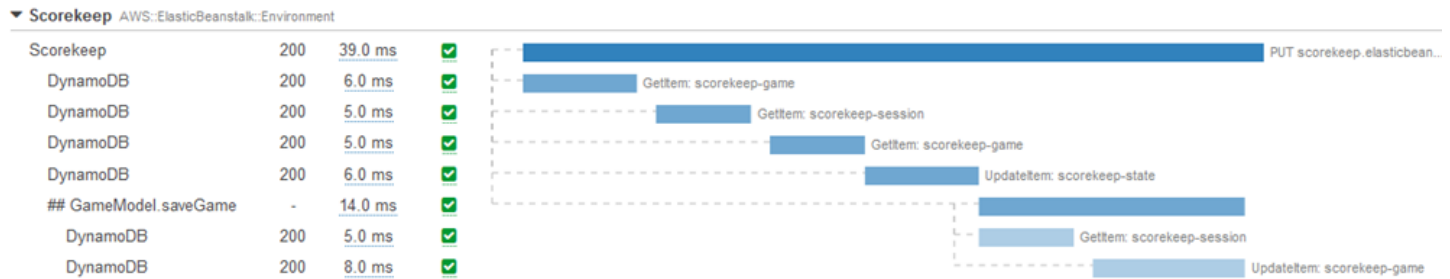
Example main.py - Subsegmen kustom

```
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)
    })
}
```

Tangkapan layar berikut menunjukkan sebuah contoh cara subsegmen saveGame mungkin muncul dalam penelusuran untuk aplikasi Scorekeep.



Menambahkan anotasi dan metadata ke segmen dengan X-Ray SDK for Go

Anda dapat mencatat informasi tambahan tentang permintaan, lingkungan, atau aplikasi Anda dengan anotasi dan metadata. Anda dapat menambahkan anotasi dan metadata ke segmen yang dibuat oleh SDK X-Ray, atau subsegmen kustom yang Anda buat.

Anotasi adalah pasangan kunci-nilai dengan string, nomor, atau nilai-nilai Boolean. Anotasi diindekskan untuk digunakan dengan [Ekspresi filter](#). Gunakan anotasi untuk mencatat data yang ingin Anda gunakan untuk mengelompokkan pelacakan di konsol tersebut, atau saat memanggil API [GetTraceSummaries](#).

Metadata adalah pasangan kunci-nilai yang dapat memiliki nilai dari setiap tipe, termasuk objek dan daftar, tetapi tidak diindekskan untuk digunakan dengan ekspresi filter. Gunakan metadata untuk mencatat data tambahan yang ingin disimpan dalam pelacakan tetapi tidak perlu digunakan dengan pencarian.

Selain anotasi dan metadata, Anda juga dapat [mencatat string ID pengguna](#) pada segmen. Pengguna IDs direkam dalam bidang terpisah pada segmen dan diindeks untuk digunakan dengan pencarian.

Bagian-bagian

- [Mencatat anotasi dengan X-Ray SDK for Go](#)
- [Mencatat metadata dengan X-Ray SDK for Go](#)
- [Merekam pengguna IDs dengan X-Ray SDK for Go](#)

Mencatat anotasi dengan X-Ray SDK for Go

Gunakan anotasi untuk mencatat informasi tentang segmen yang ingin Anda indeks untuk pencarian.

Persyaratan Anotasi

- Tombol — Kunci untuk anotasi X-Ray dapat memiliki hingga 500 karakter alfanumerik. Anda tidak dapat menggunakan spasi atau simbol selain titik atau titik (.).
- Nilai — Nilai untuk anotasi X-Ray dapat memiliki hingga 1.000 karakter Unicode.
- Jumlah Anotasi — Anda dapat menggunakan hingga 50 anotasi per jejak.

Untuk mencatat anotasi, hubungi `AddAnnotation` dengan string yang berisi metadata yang ingin Anda kaitkan dengan segmen.

```
xray.AddAnnotation(key string, value interface{})
```

SDK mencatat anotasi sebagai pasangan nilai kunci dalam objek `annotations` di dokumen segmen. Memanggil `AddAnnotation` dua kali dengan tombol yang sama menimpa nilai yang tercatat sebelumnya pada segmen yang sama.

Untuk menemukan pelacakan yang memiliki anotasi dengan nilai-nilai tertentu, gunakan `annotation[key]` kata kunci dalam [Ekspresi filter](#).

Mencatat metadata dengan X-Ray SDK for Go

Gunakan metadata untuk mencatat informasi di segmen yang tidak perlu diindekskan untuk pencarian.

Untuk mencatat metadata, panggil `AddMetadata` Dengan string yang berisi metadata yang ingin Anda kaitkan dengan segmen.

```
xray.AddMetadata(key string, value interface{})
```

Merekam pengguna IDs dengan X-Ray SDK for Go

Rekam segmen pengguna IDs berdasarkan permintaan untuk mengidentifikasi pengguna yang mengirim permintaan.

Untuk merekam pengguna IDs

1. Dapatkan referensi ke segmen saat ini dari `AWSXRay`.

```
import (
```

```
"context"  
  "github.com/aws/aws-xray-sdk-go/xray"  
)  
  
mySegment := xray.GetSegment(context)
```

2. Panggil setUser dengan ID String dari pengguna yang mengirim permintaan.

```
mySegment.User = "U12345"
```

Untuk menemukan pelacakan untuk ID pengguna, gunakan kata kunci user dalam [Ekspresi filter](#).

Bekerja dengan Java

Ada dua cara untuk instrumen aplikasi Java Anda untuk mengirim jejak ke X-Ray:

- [AWS Distro untuk OpenTelemetry Java — AWS Distribusi yang menyediakan serangkaian pustaka open source untuk mengirim metrik dan jejak yang berkorelasi ke beberapa solusi AWS pemantauan, termasuk Amazon,, AWS X-Ray dan Amazon OpenSearch Service CloudWatch, melalui Distro for Collector.AWS OpenTelemetry](#)
- [AWS X-Ray SDK for Java - Satu set perpustakaan untuk menghasilkan dan mengirim jejak ke X-Ray melalui daemon X-Ray.](#)

Untuk informasi selengkapnya, lihat [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#).

AWS Distro untuk OpenTelemetry Java

Dengan AWS Distro for OpenTelemetry (ADOT) Java, Anda dapat menginstruksikan aplikasi Anda sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan termasuk Amazon CloudWatch,, AWS X-Ray dan Amazon Service. OpenSearch Menggunakan X-Ray dengan ADOT memerlukan dua komponen: OpenTelemetry SDK diaktifkan untuk digunakan dengan X-Ray, dan AWS Distro untuk OpenTelemetry Kolektor diaktifkan untuk digunakan dengan X-Ray. ADOT Java menyertakan dukungan instrumentasi otomatis, memungkinkan aplikasi Anda mengirim jejak tanpa perubahan kode.

Untuk memulai, lihat [dokumentasi AWS Distro untuk OpenTelemetry Java](#).

Untuk informasi selengkapnya tentang menggunakan AWS Distro untuk OpenTelemetry with AWS X-Ray dan lainnya Layanan AWS, lihat [AWS Distro untuk OpenTelemetry atau Distro untuk AWS Dokumentasi](#). OpenTelemetry

Untuk informasi selengkapnya tentang dukungan dan penggunaan bahasa, lihat [AWS Observabilitas di GitHub](#).

AWS X-Ray SDK for Java

X-Ray SDK for Java adalah serangkaian pustaka untuk aplikasi web Java yang menyediakan kelas dan metode untuk menghasilkan dan mengirim pelacakan data ke daemon X-Ray. Data

pelacakan mencakup informasi tentang permintaan HTTP masuk yang disajikan oleh aplikasi, dan panggilan yang dilakukan aplikasi ke layanan hilir menggunakan AWS SDK, klien HTTP, atau konektor database SQL. Anda juga dapat membuat segmen secara manual dan menambahkan informasi debug dalam anotasi dan metadata.

X-Ray SDK for Java merupakan proyek sumber terbuka. Anda dapat mengikuti proyek dan mengirimkan masalah dan menarik permintaan di GitHub: github.com/aws/aws-xray-sdk-java

Mulai dengan [menambahkan AWSXRayServletFilter sebagai filter servlet](#) untuk pelacakan permintaan yang masuk. Filter servlet membuat [segmen](#). Ketika segmen terbuka, Anda dapat menggunakan metode klien SDK untuk menambahkan informasi ke segmen dan membuat subsegmen untuk penelusuran panggilan hilir. SDK juga secara otomatis mencatat pengecualian bahwa aplikasi Anda melempar sementara segmen terbuka.

Mulai rilis 1.3, Anda dapat instrumen aplikasi Anda menggunakan [berorientasi aspek pemrograman \(AOP\) di Spring](#). Artinya, Anda dapat menginstruksikan aplikasi Anda, saat sedang berjalan AWS, tanpa menambahkan kode apa pun ke runtime aplikasi Anda.

Selanjutnya, gunakan X-Ray SDK for Java untuk menginstrumentasikan klien AWS SDK untuk Java Anda [dengan menyertakan submodul SDK Instrumentor dalam](#) konfigurasi build Anda. Setiap kali Anda melakukan panggilan ke hilir Layanan AWS atau sumber daya dengan klien yang diinstrumentasi, SDK akan mencatat informasi tentang panggilan di subsegmen. Layanan AWS dan sumber daya yang Anda akses dalam layanan muncul sebagai node hilir pada peta jejak untuk membantu Anda mengidentifikasi kesalahan dan masalah pembatasan pada koneksi individual.

Jika Anda tidak ingin menginstrumentasikan semua panggilan hilir Layanan AWS, Anda dapat meninggalkan submodul Instrumentor dan memilih klien mana yang akan diinstrumensikan. Instrumen klien individu [dengan menambahkan TracingHandler](#) ke klien layanan AWS SDK.

Submodul X-Ray SDK for Java lainnya menyediakan instrumentasi untuk panggilan hilir ke web HTTP APIs dan database SQL. Anda dapat [menggunakan X-Ray SDK for Java dari versi HTTPClient dan HTTPClientBuilder](#) di submodul Apache HTTP untuk instrumen klien Apache HTTP. Untuk instrumen kueri SQL, [tambahkan interceptor SDK pada sumber data Anda](#).

Setelah Anda mulai menggunakan SDK, sesuaikan perilakunya dengan [mengonfigurasi filter perekam dan servlet](#). Anda dapat menambahkan plugin untuk mencatat data mengenai sumber daya komputasi yang berjalan di aplikasi Anda, menyesuaikan perilaku sampling dengan mendefinisikan aturan sampling, dan mengatur tingkat log untuk melihat lebih atau kurang informasi dari SDK dalam log aplikasi Anda.

Catat informasi tambahan tentang permintaan dan pekerjaan yang dilakukan aplikasi Anda dalam [anotasi dan metadata](#). Anotasi adalah pasangan kunci-nilai sederhana yang diindeks untuk digunakan dengan [ekspresi filter](#), sehingga Anda dapat mencari pelacakan yang berisi data tertentu. Entri metadata kurang bersifat membatasi dan dapat mencatat seluruh objek dan array — segala yang dapat disambungkan ke dalam JSON.

Anotasi dan Metadata

Anotasi dan metadata adalah teks abritari yang Anda tambahkan ke segmen dengan X-Ray SDK. Anotasi diindekskan untuk digunakan dengan ekspresi filter. Metadata tidak diindeks, tetapi dapat dilihat di segmen mentah dengan konsol X-Ray atau API. Siapa pun yang Anda berikan akses baca ke X-Ray dapat melihat data ini.

Bila Anda memiliki banyak klien diinstrumentasi dalam kode Anda, segmen permintaan tunggal dapat berisi banyak subsegmen, satu untuk setiap panggilan yang dibuat dengan klien berinstrumen. Anda dapat mengatur dan mengelompokkan subsegmen dengan membungkus panggilan klien di [subsegmen kustom](#). Anda dapat membuat subsegmen kustom untuk seluruh fungsi atau bagian dari kode apa pun, dan mencatat metadata dan anotasi pada subsegmen alih-alih menulis semuanya pada segmen induk.

Submodul

Anda dapat mengunduh X-Ray SDK for Java dari Maven. X-Ray SDK for Java dibagi menjadi submodul dengan kasus penggunaan, dengan tagihan bahan untuk manajemen versi:

- [aws-xray-recorder-sdk-core](#) (wajib) – Fungsi dasar untuk membuat segmen dan transmisi segmen. Termasuk `AWSXRayServletFilter` untuk instrumenting permintaan masuk.
- [aws-xray-recorder-sdk-aws-sdk](#)— Instrumen panggilan untuk Layanan AWS dilakukan dengan AWS SDK untuk Java klien dengan menambahkan klien penelusuran sebagai penanganan permintaan.
- [aws-xray-recorder-sdk-aws-sdk-v2](#)— Instrumen panggilan untuk Layanan AWS dilakukan dengan klien AWS SDK untuk Java 2.2 dan yang lebih baru dengan menambahkan klien penelusuran sebagai intereceptor permintaan.
- [aws-xray-recorder-sdk-aws-sdk-instrumentor](#)— Dengan `aws-xray-recorder-sdk-aws-sdk`, instrumen semua AWS SDK untuk Java klien secara otomatis.

- [aws-xray-recorder-sdk-aws-sdk-v2-instrumentor](#)— Dengan `aws-xray-recorder-sdk-aws-sdk-v2`, instrumen semua klien AWS SDK untuk Java 2.2 dan yang lebih baru secara otomatis.
- [aws-xray-recorder-sdk-apache-http](#) – Instrumen panggilan HTTP keluar yang dibuat dengan klien HTTP Apache.
- [aws-xray-recorder-sdk-spring](#) – Menyediakan pencegat untuk aplikasi Spring AOP Framework.
- [aws-xray-recorder-sdk-sql-postgres](#) – Instrumen panggilan keluar ke basis data PostgreSQL yang dibuat dengan JDBC.
- [aws-xray-recorder-sdk-sql-mysql](#) – Instrumen panggilan keluar ke basis data MySQL yang dibuat dengan JDBC.
- [aws-xray-recorder-sdk-bom](#) – Menyediakan tagihan materi yang dapat Anda gunakan untuk menentukan versi yang akan digunakan untuk semua submodul.
- [aws-xray-recorder-sdk-metrics](#)— Publikasikan CloudWatch metrik Amazon tanpa sampel dari segmen X-Ray yang Anda kumpulkan.

Jika Anda menggunakan Maven atau Gradle untuk membangun aplikasi Anda, [Menambahkan X-Ray SDK for Java ke konfigurasi pembuatan Anda](#).

Untuk dokumentasi referensi kelas SDK dan metode, lihat [AWS X-Ray Referensi API SDK for Java](#).

Persyaratan

X-Ray SDK for Java memerlukan Java 8 atau lebih baru, Servlet API 3, SDK, AWS dan Jackson.

SDK tergantung pada pustaka berikut saat kompilasi dan waktu aktif:

- AWS SDK for Java versi 1.11.398 atau yang lebih baru
- API Servlet 3.1.0

Dependensi ini dinyatakan dalam file `pom.xml` dan disertakan secara otomatis jika Anda membangun menggunakan Maven atau Gradle.

Jika Anda menggunakan pustaka yang disertakan dalam X-Ray SDK for Java, Anda harus menggunakan versi yang disertakan. Misalnya, jika Anda sudah bergantung pada Jackson saat waktu aktif dan termasuk file JAR dalam deployment Anda untuk ketergantungan itu, Anda harus menghapus file-file JAR karena SDK JAR termasuk versi sendiri dari Jackson pustaka.

Manajemen dependensi

X-Ray SDK for Java tersedia dari Maven:

- Grup - `com.amazonaws`
- Artifact – `aws-xray-recorder-sdk-bom`
- Versi - `2.11.0`

Jika Anda menggunakan Maven untuk membangun aplikasi Anda, menambahkan SDK sebagai ketergantungan di file `pom.xml` Anda.

Example `pom.xml` - dependensi

```
<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>

```

Untuk Gradle, tambahkan SDK sebagai dependensi waktu kompilasi di file `build.gradle`.

Example `build.gradle` – dependensi

```

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')
    }
}

```

Jika Anda menggunakan Elastic Beanstalk untuk mendeploy aplikasi Anda, Anda dapat menggunakan Maven atau Gradle untuk membangun on-instans setiap kali Anda mendeploy, alih-alih membangun dan mengunggah arsip besar yang mencakup semua dependensi Anda. Lihat [aplikasi sampel](#) sebagai contoh yang menggunakan Gradle.

AWS X-Ray agen instrumentasi otomatis Java

Agen AWS X-Ray instrumentasi otomatis untuk Java adalah solusi penelusuran yang menginstruksikan aplikasi web Java Anda dengan upaya pengembangan minimal. Agen memungkinkan pelacakan untuk aplikasi berbasis servlet dan semua permintaan hilir agen dibuat

dengan dukungan kerangka kerja dan pustaka. Ini termasuk permintaan HTTP Apache hilir, permintaan AWS SDK, dan kueri SQL dibuat menggunakan driver JDBC. Agen menyebarkan konteks X-Ray, termasuk semua segmen dan subsegmen aktif, di seluruh utas. Semua konfigurasi dan fleksibilitas X-Ray SDK masih tersedia dengan agen Java. Default yang sesuai dipilih untuk memastikan bahwa agen bekerja dengan sedikit usaha.

Solusi agen X-Ray paling cocok untuk server aplikasi web Java yang berbasis servlet. Jika aplikasi Anda menggunakan kerangka kerja asinkron, atau tidak dimodelkan dengan baik sebagai layanan respon permintaan, Anda mungkin ingin mempertimbangkan instrumentasi manual dengan SDK sebagai gantinya.

Agen X-Ray dibangun menggunakan toolkit Pemahaman Sistem Terdistribusi, atau Di. SCo Di SCo adalah kerangka open source untuk membangun agen Java yang dapat digunakan dalam sistem terdistribusi. Meskipun tidak perlu memahami Di SCo untuk menggunakan agen X-Ray, Anda dapat mempelajari lebih lanjut tentang proyek ini dengan mengunjungi [beranda di GitHub](#). Agen X-Ray juga sepenuhnya bersumber terbuka. Untuk melihat kode sumber, memberikan kontribusi, atau mengangkat masalah tentang agen, kunjungi [repositorinya](#) di GitHub.

Aplikasi sampel

Aplikasi [eb-java-scorekeep](#) sampel disesuaikan untuk diinstrumentasi dengan agen X-Ray. Cabang ini tidak berisi filter servlet atau konfigurasi pencatat, karena fungsi-fungsi ini dilakukan oleh agen. Untuk menjalankan aplikasi lokal atau menggunakan sumber daya AWS, ikuti langkah-langkah dalam file readme aplikasi sampel. Petunjuk untuk menggunakan aplikasi sampel untuk menghasilkan penelusuran X-Ray ada pada [contoh tutorial aplikasi](#).

Memulai

Untuk memulai dengan X-Ray agen instrumentasi otomatis Java pada aplikasi Anda sendiri, ikuti langkah-langkah ini.

1. Jalankan daemon X-Ray di lingkungan Anda. Untuk informasi selengkapnya, lihat [AWS X-Ray daemon](#).
2. Unduh [distribusi agen terbaru](#). Unzip arsip dan perhatikan lokasinya dalam sistem file Anda. Isinya akan terlihat seperti berikut.

```
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. Modifikasi argumen JVM aplikasi Anda untuk memasukkan hal-hal berikut, yang memungkinkan agen. Pastikan argumen `-javaagent` ditempatkan sebelum argumen `-jar` jika berlaku. Proses untuk memodifikasi argumen JVM bervariasi tergantung pada alat dan kerangka kerja yang Anda gunakan untuk meluncurkan server Java Anda. Konsultasikan dokumentasi kerangka kerja server Anda untuk panduan khusus.

```
-javaagent:/<path-to-disco>/disco-java-agent.jar=pluginPath=/<path-to-disco>/disco-plugins
```

4. Untuk menentukan bagaimana nama aplikasi Anda muncul di konsol X-Ray, atur variabel `AWS_XRAY_TRACING_NAME` lingkungan atau properti `com.amazonaws.xray.strategy.tracingName` sistem. Jika tidak ada nama yang diberikan, nama default digunakan.
5. Mulai ulang server atau kontainer Anda. Permintaan masuk dan panggilan hilir mereka sekarang ditelusuri. Jika Anda tidak melihat hasil yang diharapkan, lihat [the section called “Pemecahan Masalah”](#).

Konfigurasi

Agen X-Ray dikonfigurasi oleh file JSON eksternal yang disediakan pengguna. Secara default, file ini adalah akar dari classpath pengguna (misalnya, di `resources` direktori) bernama `xray-agent.json`. Anda dapat mengonfigurasi lokasi kustom untuk file konfigurasi dengan menetapkan sistem properti `com.amazonaws.xray.configFile` menuju sistem file jalur absolut dari file konfigurasi Anda.

Contoh file konfigurasi ditampilkan nanti.

```
{
  "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
}

```

Spesifikasi konfigurasi

Tabel berikut mencakup nilai-nilai yang valid untuk setiap bidang. Nama properti merupakan kasus sensitif, namun tidak untuk kuncinya. Untuk properti yang dapat diganti oleh variabel lingkungan dan properti sistem, urutan prioritas selalu variabel lingkungan, lalu properti sistem, dan kemudian file konfigurasi. Untuk informasi tentang properti yang dapat Anda ganti, lihat [Variabel-variabel lingkungan](#). Semua bidang bersifat opsional.

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
serviceName	String	Semua string	Nama layanan terinstrumentasi Anda karena akan muncul di konsol X-Ray.	AWS_XRAY_TRACING_NAME	com.amazonaws.xray.strategy.tracingName	XRayInstrumentedService
contextMissingStrategy	String	LOG_ERROR, IGNORE_ERROR	Tindakan yang diambil oleh agen ketika mencoba untuk	AWS_XRAY_CONTEXT_MISSING	com.amazonaws.xray.strategy.contextMissingStrategy	LOG_ERROR

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
			menggunakan konteks segmen X-Ray tetapi tidak ada yang hadir.			
daemonAddress	String	Alamat IP dan port terformat, atau daftar alamat TCP dan UDP	Alamat yang digunakan agen untuk berkomunikasi dengan daemon X-Ray.	AWS_XRAY_DAEMON_ADDRESS	com.amazonaws.xray.emitter.daemonAddress	127.0.0.1:2000
tracingEnabled	Boolean	Betul, Salah	Mengaktifkan instrumenasi oleh agen X-Ray.	AWS_XRAY_TRACING_ENABLED	com.amazonaws.xray.tracingEnabled	BETUL

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
samplingStrategy	String	PUSAT, LOKAL, TIDAK ADA, SEMUA	Strategi sampling yang digunakan oleh agen. SEMUA menangkap semua permintaan, TIDAK ADA menangkap tidak ada permintaan. Lihat aturan pengambilan sampel .	N/A	N/A	PUSAT
traceInjectionAwalan	String	Semua string	Termasuk awalan yang disediakan sebelum jejak yang disuntikkan IDs di log.	N/A	N/A	Tidak ada (string kosong)

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
samplingRulesManifest	String	Path file absolut	Path pada file aturan sampling kustom yang akan digunakan sebagai sumber aturan sampling bagi strategi pengambilan sampel lokal, atau aturan mundur untuk strategi pusat.	N/A	N/A	DefaultSamplingRules.json

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
awsServiceHandlerManifestasi	String	Path file absolut	Path menuju parameter kustom mengizinkan daftar, yang menangkap informasi tambahan dari klien AWS SDK.	N/A	N/A	DefaultOperationParameterWhitelist.json
awsSdkVersion	Bilangan Bulat	1, 2	Versi AWS SDK for Java yang Anda gunakan. Diabaikan jika awsServiceHandlerManifestasi tidak diatur juga.	N/A	N/A	2

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
maxStackTraceLength	Bilangan Bulat	Bilangan bulat non-negatif	Baris maksimum penelusuran tumpukan untuk mencatat dalam penelusuran.	N/A	N/A	50
streamingThreshold	Bilangan Bulat	Bilangan bulat non-negatif	Setelah setidaknya a banyak subsegmen ini ditutup, mereka dialirkan ke daemon out-of-band untuk menghindari potongan yang terlalu besar.	N/A	N/A	100

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
tracedInjection	Boolean	Betul, Salah	Mengaktifkan injeksi ID penelusuran X-Ray ke log jika dependensi dan konfigurasi yang dijelaskan dalam konfigurasi pencatatan juga ditambahkan. Jika tidak, tidak melakukan apa-apa.	N/A	N/A	BETUL

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
pluginsEnabled	Boolean	Betul, Salah	Mengaktifkan plugin yang merekam metadata tentang AWS lingkungan tempat Anda beroperasi. Lihat plugin .	N/A	N/A	BETUL
collectSqlQueries	Boolean	Betul, Salah	Mencatat string kueri SQL di subsegment SQL dengan basis upaya-terbaik.	N/A	N/A	SALAH

Nama properti	Tipe	Nilai yang valid	Deskripsi	Variabel Lingkungan	Properti sistem	Default
contextPropagation	Boolean	Betul, Salah	Secara otomatis menyebarkan konteks X-Ray antara thread jika benar. Jika tidak, menggunakan Thread Local untuk menyimpan konteks dan propagasi manual di seluruh utas diperlukan.	N/A	N/A	BETUL

Konfigurasi log

Tingkat log agen X-Ray dapat dikonfigurasi dengan cara yang sama seperti X-Ray SDK for Java. Lihat [Pencatatan log](#) untuk informasi selengkapnya tentang mengonfigurasi log dengan X-Ray SDK for Java.

Instrumentasi manual

Jika Anda ingin melakukan instrumentasi manual selain agen instrumentasi otomatis, tambahkan SDK X-Ray sebagai sebuah dependensi pada proyek Anda. Perhatikan bahwa filter servlet kustom SDK disebutkan dalam [Menelusuri Permintaan Masuk](#) tidak kompatibel dengan agen X-Ray.

 Note

Anda harus menggunakan versi terbaru SDK X-Ray untuk melakukan instrumentasi manual saat menggunakan agen juga.

Jika Anda bekerja di sebuah proyek Maven, tambahkan dependensi berikut untuk file `pom.xml` Anda.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
    <version>2.11.0</version>
  </dependency>
</dependencies>
```

Jika Anda bekerja di sebuah proyek Maven, tambahkan dependensi berikut untuk file `build.gradle` Anda.

```
implementation 'com.amazonaws:aws-xray-recorder-sdk-core:2.11.0'
```

Anda dapat menambahkan [subsegmen khusus](#) selain [anotasi, metadata, dan pengguna IDs](#) saat menggunakan agen, seperti yang Anda lakukan dengan SDK normal. Agen secara otomatis menyebarkan konteks di thread, sehingga tidak ada workarounds untuk menyebarkan konteks yang diperlukan ketika bekerja dengan aplikasi multithreaded.

Pemecahan Masalah

Karena agen menawarkan instrumentasi otomatis sepenuhnya, akan sulit untuk mengidentifikasi akar penyebab masalah ketika Anda mengalami masalah. Jika agen X-Ray tidak bekerja seperti yang Anda harapkan, tinjau masalah dan solusi berikut ini. Agen X-Ray dan SDK menggunakan Jakarta Commons Logging (JCL). Untuk melihat output logging, pastikan bahwa jembatan yang menghubungkan JCL ke backend logging Anda ada di classpath, seperti pada contoh berikut: atau. `log4j-jcl jcl-over-slf4j`

Masalah: Saya telah mengaktifkan agen Java pada aplikasi saya tetapi tidak ada apa pun di konsol X-Ray

Apakah daemon X-Ray berjalan pada mesin yang sama?

Jika tidak, lihat [Dokumentasi daemon X-Ray](#) untuk mengaturnya.

Di log aplikasi Anda, apakah Anda melihat pesan seperti “Inisialisasi perekam agen X-Ray”?

Jika Anda telah menambahkan agen dengan benar ke aplikasi Anda, pesan ini dicatat pada tingkat INFO saat aplikasi Anda dimulai, sebelum mulai menerima permintaan. Jika pesan ini tidak ada, maka agen Java tidak berjalan dengan proses Java Anda. Pastikan Anda telah mengikuti semua langkah pengaturan dengan benar tanpa adanya kesalahan ketik.

Di log aplikasi Anda, apakah Anda melihat beberapa pesan kesalahan yang mengatakan sesuatu seperti “Menekan AWS X-Ray konteks yang hilang pengecualian”?

Kesalahan ini terjadi karena agen mencoba untuk instrumen permintaan hilir, seperti permintaan AWS SDK atau kueri SQL, tetapi agen tidak dapat secara otomatis membuat segmen. Jika Anda melihat banyak kesalahan ini, agen mungkin bukan alat terbaik untuk kasus penggunaan Anda dan Anda mungkin ingin mempertimbangkan instrumentasi manual dengan SDK X-Ray sebagai gantinya. Selain itu, Anda dapat mengaktifkan X-Ray SDK [debug log](#) untuk melihat pelacakan tumpukan tempat pengecualian konteks hilang terjadi. Anda dapat membungkus bagian-bagian kode Anda dengan segmen kustom, yang harus menyelesaikan kesalahan ini. Untuk contoh membungkus permintaan hilir dengan segmen kustom, lihat kode sampel di [instrumenting kode startup](#).

Masalah: Beberapa segmen yang saya harapkan tidak muncul di konsol X-Ray

Apakah aplikasi Anda menggunakan multithreading?

Jika beberapa segmen yang Anda harapkan untuk dibuat tidak muncul di konsol Anda, latar belakang thread dalam aplikasi Anda mungkin menjadi penyebabnya. Jika aplikasi Anda melakukan tugas menggunakan utas latar belakang yang “fire and forget”, seperti melakukan panggilan satu kali ke fungsi Lambda dengan AWS SDK, atau melakukan polling beberapa titik akhir HTTP secara berkala, itu mungkin membingungkan agen saat menyebarkan konteks di seluruh utas. Untuk memverifikasi masalah Anda ini, aktifkan log debug X-Ray SDK dan periksa pesan seperti: Tidak memancarkan segmen bernama <NAME > seperti induknya di-progress subsegment. Untuk mengatasi hal ini, Anda dapat mencoba bergabung dengan thread latar belakang sebelum server Anda kembali untuk memastikan semua pekerjaan yang dilakukan di dalamnya dicatat. Atau, Anda dapat mengatur konfigurasi `contextPropagation` agen ke `false` untuk menonaktifkan propagasi konteks di thread latar belakang. Jika Anda melakukan ini, Anda harus secara manual melakukan instrumen utas tersebut dengan segmen kustom atau mengabaikan konteks pengecualian yang hilang yang mereka hasilkan.

Sudahkah Anda mengatur aturan pengambilan sampel?

Jika segmen yang tampak acak atau tidak terduga muncul di konsol X-Ray, atau segmen yang Anda harapkan berada di konsol, Anda mungkin mengalami masalah pengambilan sampel. Agen X-Ray menerapkan pengambilan sampel terpusat ke semua segmen yang dibuatnya, menggunakan aturan dari konsol X-Ray. Aturan default adalah 1 segmen per detik, ditambah 5% dari segmen sesudahnya, dijadikan sampel. Ini berarti segmen yang dibuat cepat dengan agen mungkin tidak dijadikan sampel. Untuk mengatasi masalah ini, Anda harus membuat aturan pengambilan sampel kustom di konsol X-Ray yang secara tepat mencoba segmen yang diinginkan. Untuk informasi selengkapnya, lihat [sampling](#).

Mengonfigurasi X-Ray SDK for Java

X-Ray SDK for Java mencakup kelas bernama `AWSEXRay` yang menyediakan pencatat global. Ini adalah `TracingHandler` yang dapat Anda gunakan untuk menginstrumentasikan kode Anda. Anda dapat mengonfigurasi pencatat global untuk menyesuaikan `AWSEXRayServletFilter` yang membuat segmen untuk panggilan HTTP masuk.

Bagian-bagian

- [Plugin layanan](#)
- [Aturan pengambilan sampel](#)
- [Pencatatan log](#)
- [Listener segmen](#)
- [Variabel-variabel lingkungan](#)
- [Properti sistem](#)

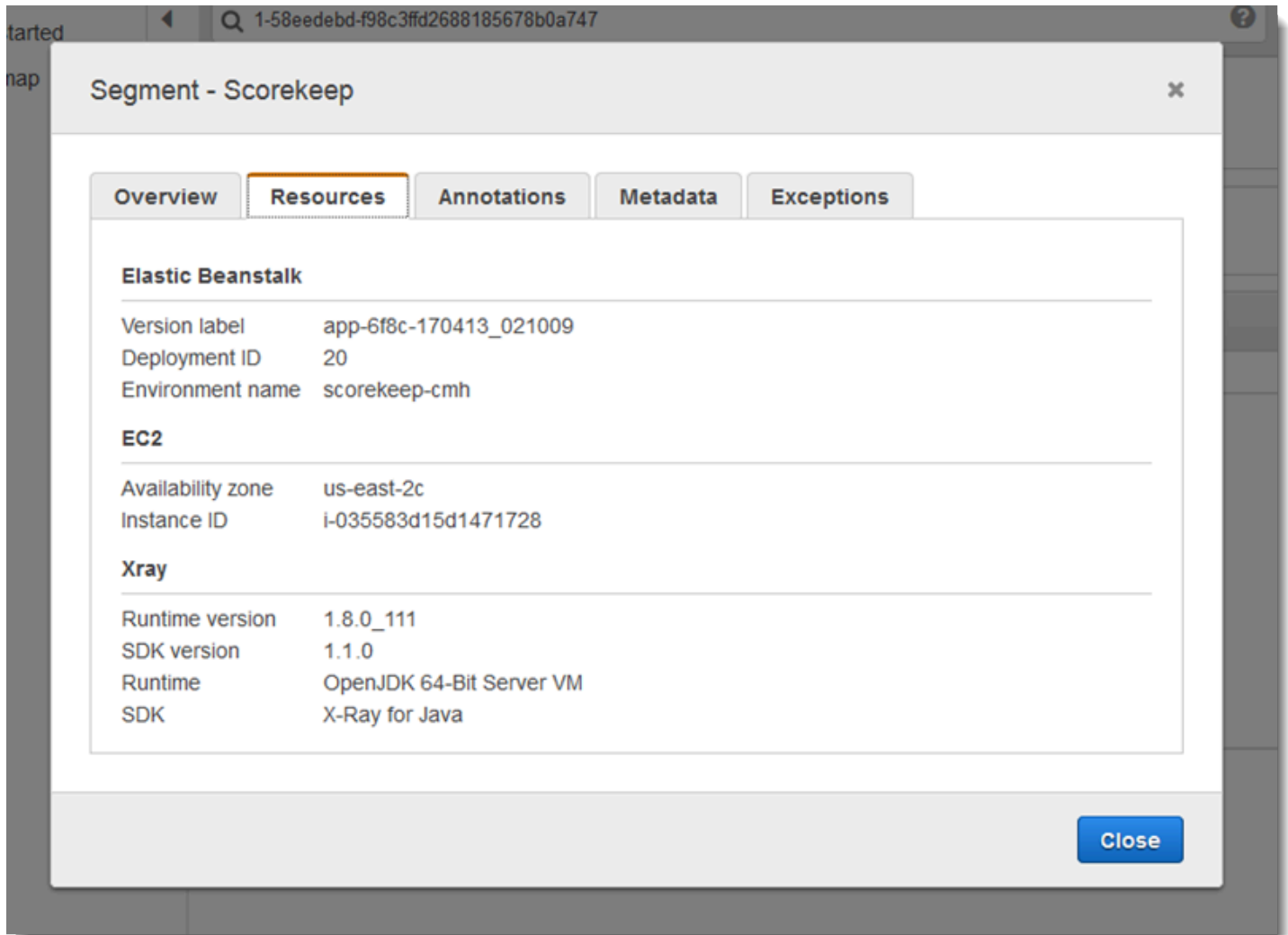
Plugin layanan

Gunakan `plugins` untuk mencatat informasi tentang layanan yang meng-hosting aplikasi Anda.

Plugin

- Amazon EC2 — `EC2Plugin` menambahkan ID instans, Availability Zone, dan Grup CloudWatch Log.
- Elastic Beanstalk – `ElasticBeanstalkPlugin` menambahkan nama lingkungan, label versi, dan ID deployment.
- Amazon ECS – `ECSPPlugin` menambahkan ID kontainer.

- Amazon EKS — EKSPugin menambahkan ID kontainer, nama cluster, ID pod, dan Grup CloudWatch Log.



Untuk menggunakan plugin, hubungi `withPlugin` di `AWSXRayRecorderBuilder` Anda.

Example `src/main/java/scorekeep/WebConfig.java` - perekam

```
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());
}
}
```

SDK juga menggunakan pengaturan plugin untuk mengatur bidang origin pada segmen. Ini menunjukkan jenis AWS sumber daya yang menjalankan aplikasi Anda. Saat Anda menggunakan beberapa plugin, SDK menggunakan urutan resolusi berikut untuk menentukan asal: ElasticBeanstalk > EKS > ECS >. EC2

Aturan pengambilan sampel

SDK menggunakan aturan pengambilan sampel yang Anda tetapkan di konsol X-Ray untuk menentukan permintaan yang akan dicatat. Aturan default menelusuri permintaan pertama setiap detik, dan lima persen permintaan tambahan di semua layanan yang mengirim pelacakan ke X-Ray. [Buat aturan tambahan di konsol X-Ray](#) untuk menyesuaikan jumlah data yang dicatat untuk setiap aplikasi Anda.

SDK menerapkan aturan kustom sesuai urutan penetapannya. Jika permintaan cocok dengan beberapa aturan kustom, SDK hanya menerapkan aturan pertama.

Note

Jika SDK tidak dapat mencapai X-Ray untuk mendapatkan aturan pengambilan sampel, SDK akan beralih ke aturan lokal default dari permintaan pertama setiap detik, dan lima persen permintaan tambahan per host. Hal ini dapat terjadi jika host tidak memiliki izin untuk memanggil sampling APIs, atau tidak dapat terhubung ke daemon X-Ray, yang bertindak sebagai proxy TCP untuk panggilan API yang dilakukan oleh SDK.

Anda juga dapat mengonfigurasi SDK untuk memuat aturan sampling dari dokumen JSON. SDK dapat menggunakan aturan lokal sebagai cadangan jika terjadi kasus tidak dapat mengambil sampel X-Ray, atau menggunakan aturan lokal secara eksklusif.

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
  }
}
```

Contoh ini menentukan satu aturan kustom dan aturan default. Aturan kustom menerapkan tingkat pengambilan sampel lima persen tanpa jumlah minimum permintaan untuk melacak jalur di `/api/move/`. Aturan default menelusuri permintaan pertama setiap detik dan 10 persen dari permintaan tambahan.

Kerugian dari menentukan aturan secara lokal adalah bahwa target tetap diterapkan oleh setiap instans pencatat secara independen, alih-alih dikelola oleh layanan X-Ray. Ketika Anda men-deploy lebih banyak host, laju tetap akan dikalikan, sehingga sulit untuk mengontrol jumlah data yang dicatat.

AWS Lambda Aktif, Anda tidak dapat mengubah laju pengambilan sampel. Jika fungsi Anda dipanggil oleh layanan yang diinstrumentasikan, panggilan yang menghasilkan permintaan yang sampelnya diambil oleh layanan yang akan dicatat oleh Lambda. Jika pelacakan aktif diaktifkan dan tidak ada header pelacakan, Lambda membuat keputusan pengambilan sampel.

Untuk memberikan aturan cadangan di Spring, konfigurasi pencatat global dengan `CentralizedSamplingStrategy` di kelas konfigurasi.

Example `src/main/java/myapp/WebConfig.java` - konfigurasi perekam

```
import com.amazonaws.xray.AWSXRay;
```

```
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.java.x.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());
    }
}
```

Untuk Tomcat, tambahkan listener yang memperluas `ServletContextListener` dan mendaftarkan listener di pendeskripsi deployment.

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>

```

Untuk menggunakan aturan lokal saja, ganti `CentralizedSamplingStrategy` dengan `LocalizedSamplingStrategy`.

```
builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));
```

Pencatatan log

Secara default, SDK menghasilkan pesan tingkat `ERROR` untuk log aplikasi Anda. Anda dapat mengaktifkan pencatatan tingkat debug di SDK untuk menghasilkan log lebih terperinci ke berkas log aplikasi Anda. Tingkat log yang valid adalah `DEBUG`, `INFO`, `WARN`, `ERROR`, dan `FATAL`. Tingkat log `FATAL` mengheningkan semua pesan log karena SDK tidak mencatat di tingkat fatal.

Example application.properties

Atur tingkat pencatatan dengan properti `logging.level.com.amazonaws.xray`.

```
logging.level.com.amazonaws.xray = DEBUG
```

Gunakan log debug untuk mengidentifikasi masalah, seperti subsegmen yang tidak tertutup, saat Anda [menghasilkan subsegmen secara manual](#).

Injeksi ID pelacakan ke log

Untuk mengekspos ID pelacakan yang memenuhi syarat ke pernyataan log Anda, Anda dapat menyuntikkan ID ke dalam konteks diagnostik yang dipetakan (MDC). Menggunakan antarmuka `SegmentListener`, metode dipanggil dari pencatat X-Ray selama peristiwa siklus hidup segmen. Ketika segmen atau subsegmen dimulai, ID pelacakan yang memenuhi syarat disuntikkan ke MDC

dengan kunci `AWS-XRAY-TRACE-ID`. Ketika segmen tersebut berakhir, kunci akan dihapus dari MDC. Hal ini memperlihatkan ID pelacakan ke pustaka pencatatan yang digunakan. Ketika subsegmen berakhir, ID induknya disuntikkan ke MDC.

Example ID pelacakan yang memenuhi syarat

ID yang memenuhi syarat direpresentasikan sebagai `TraceID@EntityID`

```
1-5df42873-011e96598b447dfca814c156@541b3365be3dafc3
```

Fitur ini bekerja dengan aplikasi Java yang diinstrumentasi dengan AWS X-Ray SDK for Java, dan mendukung konfigurasi logging berikut:

- SLF4J front-end API dengan backend Logback
- SLF4API front-end J dengan backend Log4J2
- API front-end Log4J2 dengan backend Log4J2

Lihat tab berikut untuk kebutuhan setiap front end dan backend.

SLF4J Frontend

1. Tambahkan dependensi Maven berikut ke proyek Anda:

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-xray-recorder-sdk-slf4j</artifactId>
  <version>2.11.0</version>
</dependency>
```

2. Sertakan metode `withSegmentListener` ketika membangun `AWSXRayRecorder`. Ini menambahkan `SegmentListener` kelas, yang secara otomatis menyuntikkan jejak baru IDs ke SLF4 J MDC.

`SegmentListener` mengambil string opsional sebagai parameter untuk mengonfigurasi prefiks pernyataan log. Prefiks dapat dikonfigurasi dengan cara berikut:

- Tidak ada – Menggunakan prefiks `AWS-XRAY-TRACE-ID` default.
- Kosong - Menggunakan string kosong (misalnya `""`).
- Kustom - Menggunakan prefiks kustom sebagaimana ditentukan dalam string.

Example Pernyataan **AWSXRayRecorderBuilder**

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
    .standard().withSegmentListener(new SLF4JSegmentListener("CUSTOM-
    PREFIX"));
```

Log4J2 front end

1. Tambahkan dependensi Maven berikut ke proyek Anda:

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-xray-recorder-sdk-log4j</artifactId>
  <version>2.11.0</version>
</dependency>
```

2. Sertakan metode `withSegmentListener` ketika membangun `AWSXRayRecorder`. Ini akan menambahkan `SegmentListener` kelas, yang secara otomatis menyuntikkan jejak baru yang sepenuhnya memenuhi syarat IDs ke SLF4 J MDC.

`SegmentListener` mengambil string opsional sebagai parameter untuk mengonfigurasi prefiks pernyataan log. Prefiks dapat dikonfigurasi dengan cara berikut:

- Tidak ada – Menggunakan prefiks AWS-XRAY-TRACE-ID default.
- Kosong - Menggunakan string kosong (misalnya "") dan menghapus prefiks.
- Kustom - Menggunakan prefiks kustom sebagaimana ditentukan dalam string.

Example Pernyataan **AWSXRayRecorderBuilder**

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
    .standard().withSegmentListener(new Log4JSegmentListener("CUSTOM-
    PREFIX"));
```

Logback backend

Untuk memasukkan ID pelacakan ke log acara Anda, Anda harus mengubah `PatternLayout` pencatat, yang membuat format setiap pernyataan pencatatan.

1. Temukan tempat `patternLayout` dikonfigurasi. Anda dapat melakukan ini secara terprogram, atau melalui file konfigurasi XML. Untuk mempelajari selengkapnya, lihat [Konfigurasi Longback](#).
2. Sisipkan `%X{AWS-XRAY-TRACE-ID}` dimanapun di `patternLayout` untuk memasukkan ID pelacakan dalam laporan pencatatan mendatang. `%X{}` menunjukkan bahwa Anda mengambil nilai dengan kunci yang disediakan dari MDC. Untuk mempelajari lebih lanjut tentang `PatternLayouts` Logback, lihat [PatternLayout](#).

Log4J2 backend

1. Temukan tempat `patternLayout` dikonfigurasi. Anda dapat melakukan ini secara terprogram, atau melalui file konfigurasi yang ditulis dalam format XML, JSON, YAML, atau properti.

Untuk mempelajari selengkapnya tentang mengonfigurasi Log4J2 melalui file konfigurasi, lihat [Konfigurasi](#).

Untuk mempelajari selengkapnya tentang mengonfigurasi Log4J2 secara terprogram, lihat [Konfigurasi Terprogram](#).

2. Sisipkan `%X{AWS-XRAY-TRACE-ID}` dimanapun di `PatternLayout` untuk memasukkan ID pelacakan dalam laporan pencatatan mendatang. `%X{}` menunjukkan bahwa Anda mengambil nilai dengan kunci yang disediakan dari MDC. [Untuk mempelajari selengkapnya PatternLayouts di Log4J2, lihat Pattern Layout](#).

Contoh Injeksi ID pelacakan

Berikut menunjukkan string `PatternLayout` yang diubah untuk menyertakan pelacakan ID. ID pelacakan dicetak setelah nama thread (`%t`) dan sebelum tingkat log (`%-5p`).

Example **PatternLayout** Dengan Injeksi ID

```
%d{HH:mm:ss.SSS} [%t] %X{AWS-XRAY-TRACE-ID} %-5p %m%n
```

AWS X-Ray secara otomatis mencetak kunci dan ID jejak dalam pernyataan log untuk memudahkan penguraian. Berikut ini menunjukkan pernyataan log menggunakan `PatternLayout` yang diubah.

Example Pernyataan log dengan injeksi 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
```

Pesan pencatatan itu sendiri ditempatkan dalam pola `%mdn` dan diatur saat memanggil pencatat.

Listener segmen

Pendengar segmen adalah antarmuka untuk mencegah peristiwa siklus hidup seperti awal dan akhir segmen yang dihasilkan oleh `AWSXRayRecorder`. Pelaksanaan fungsi peristiwa listener segmen mungkin bertujuan untuk menambahkan anotasi yang sama untuk semua subsegmen ketika dibuat dengan [onBeginSubsegment](#), mencatat pesan setelah setiap segmen dikirim ke daemon menggunakan [afterEndSegment](#), atau untuk mencatat kueri yang dikirim oleh penghalang SQL menggunakan [beforeEndSubsegment](#) untuk memverifikasi subsegmen sebagai perwakilan kueri SQL, menambahkan metadata tambahan jika memang demikian.

Untuk melihat daftar lengkap fungsi `SegmentListener`, kunjungi dokumentasi untuk [API SDK for Java Pencatat AWS X-Ray](#).

Contoh berikut menunjukkan cara menambahkan anotasi yang konsisten untuk semua subsegmen pada pembuatan dengan [onBeginSubsegment](#) dan untuk mencetak pesan log di akhir setiap segmen dengan [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());
}
}
```

Listener segmen kustom ini kemudian direferensikan ketika membangun `AWSXRayRecorder`.

Example `AWSXRayRecorderBuilder` pernyataan

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
    .standard().withSegmentListener(new MySegmentListener());
```

Variabel-variabel lingkungan

Anda dapat menggunakan variabel lingkungan untuk mengonfigurasi X-Ray SDK for Java. SDK mendukung variabel berikut.

- `AWS_XRAY_CONTEXT_MISSING`— Setel `RUNTIME_ERROR` untuk melempar pengecualian saat kode instrumentasi Anda mencoba merekam data saat tidak ada segmen yang terbuka.

Nilai Valid

- `RUNTIME_ERROR`— Lempar pengecualian runtime.
- `LOG_ERROR`— Log kesalahan dan lanjutkan (default).
- `IGNORE_ERROR`— Abaikan kesalahan dan lanjutkan.

Kesalahan yang berkaitan dengan segmen atau subsegmen yang hilang dapat terjadi ketika Anda mencoba untuk menggunakan klien yang diinstrumentasi dalam kode perusahaan rintisan yang berjalan ketika tidak ada permintaan terbuka, atau dalam kode yang memunculkan thread baru.

- `AWS_XRAY_DAEMON_ADDRESS` – Mengatur host dan port pendengar daemon X-Ray. Secara default, SDK menggunakan `127.0.0.1:2000` untuk data pelacakan (UDP) dan pengambilan sampel (TCP). Gunakan variabel ini jika Anda telah mengonfigurasi daemon untuk [mendengarkan di port berbeda](#) atau jika berjalan pada host yang berbeda.

format

- Port yang sama – *address:port*
- Port yang berbeda – `tcp:address:port` `udp:address:port`

- `AWS_LOG_GROUP`— Tetapkan nama grup log ke grup log yang terkait dengan aplikasi Anda. Jika grup log Anda menggunakan AWS akun dan wilayah yang sama dengan aplikasi Anda, X-Ray akan secara otomatis mencari data segmen aplikasi Anda menggunakan grup log yang ditentukan ini. Untuk informasi selengkapnya tentang grup log, lihat [Bekerja dengan grup log dan aliran](#).
- `AWS_XRAY_TRACING_NAME` – Menetapkan nama layanan yang digunakan SDK untuk segmen. Menimpa nama layanan yang Anda tetapkan pada [strategi penamaan segmen](#) filter servlet.

Variabel lingkungan mengesampingkan [Properti sistem](#) yang setara dan nilai yang ditetapkan dalam kode.

Properti sistem

Anda dapat menggunakan properti sistem sebagai alternatif khusus JVM bagi [Variabel lingkungan](#). File mendukung properti berikut:

- `com.amazonaws.xray.strategy.tracingName` – Setara dengan `AWS_XRAY_TRACING_NAME`.
- `com.amazonaws.xray.emitters.daemonAddress` – Setara dengan `AWS_XRAY_DAEMON_ADDRESS`.
- `com.amazonaws.xray.strategy.contextMissingStrategy` – Setara dengan `AWS_XRAY_CONTEXT_MISSING`.

Jika kedua properti sistem dan variabel lingkungan setara ditetapkan, nilai variabel lingkungan digunakan. Salah satu metode menimpa nilai yang diatur dalam kode.

Menelusuri permintaan yang masuk dengan X-Ray SDK for Java

Anda dapat menggunakan X-Ray SDK untuk melacak permintaan HTTP masuk yang dilayani aplikasi Anda pada EC2 instans di Amazon EC2 AWS Elastic Beanstalk, atau Amazon ECS.

Gunakan `Filter` untuk menginstrumentasikan permintaan HTTP yang masuk. Ketika Anda menambahkan filter servlet X-Ray ke aplikasi Anda, X-Ray SDK for Java membuat segmen untuk setiap permintaan sampel. Segmen ini mencakup waktu, metode, dan disposisi permintaan HTTP. Instrumentasi tambahan membuat subsegmen pada segmen ini.

Note

Untuk AWS Lambda fungsi, Lambda membuat segmen untuk setiap permintaan sampel. Untuk informasi selengkapnya, lihat [AWS Lambda dan AWS X-Ray](#).

Setiap segmen memiliki nama yang mengidentifikasi aplikasi Anda dalam peta layanan. Segmen dapat diberi nama secara statis, atau Anda dapat mengonfigurasi SDK untuk nama itu secara dinamis berdasarkan header host dalam permintaan masuk. Penamaan dinamis memungkinkan Anda mengelompokkan pelacakan berdasarkan nama domain dalam permintaan, dan menerapkan nama default jika nama tersebut tidak cocok dengan pola yang diharapkan (misalnya, jika header host ditiru).

Permintaan yang Diteruskan

Jika penyeimbang beban atau perantara lainnya meneruskan permintaan ke aplikasi Anda, X-Ray akan mengambil IP klien dari header `X-Forwarded-For` dalam permintaan bukan dari sumber IP dalam paket IP. IP klien yang dicatat untuk permintaan yang diteruskan dapat ditiru, sehingga tidak dapat dipercaya.

Ketika permintaan diteruskan, SDK menetapkan bidang tambahan di segmen untuk menunjukkan ini. Jika segmen yang berisi bidang `x_forwarded_for` ditetapkan ke `true`, IP klien diambil dari header `X-Forwarded-For` dalam permintaan HTTP.

Penangan pesan membuat segmen untuk setiap permintaan masuk dengan blok `http` yang berisi informasi berikut:

- Metode HTTP – DAPATKAN, POSTING, LETAKKAN, HAPUS, dll.
- Alamat klien – Alamat IP klien yang mengirim permintaan.
- Kode respons – Kode respons HTTP untuk permintaan yang selesai.
- Timing – Waktu mulai (saat permintaan diterima) dan waktu akhir (saat respons dikirim).
- Agen pengguna — `user-agent` dari permintaan.
- Panjang konten — `content-length` dari respons.

Bagian-bagian

- [Menambahkan filter penelusuran ke aplikasi Anda \(Tomcat\)](#)
- [Menambahkan filter penelusuran ke aplikasi Anda \(spring\)](#)
- [Mengonfigurasi strategi penamaan segmen](#)

Menambahkan filter penelusuran ke aplikasi Anda (Tomcat)

Untuk Tomcat, tambahkan `<filter>` ke file proyek `web.xml`. Gunakan parameter `fixedName` untuk menentukan [nama layanan](#) untuk diterapkan ke segmen yang dibuat untuk permintaan masuk.

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>
```

Menambahkan filter penelusuran ke aplikasi Anda (spring)

Untuk Spring, tambahkan `Filter` ke kelas `WebConfig` Anda. Teruskan nama segmen ke konstruktor [AWSXRayServletFilter](#) sebagai string.

Example `src/main/java/myapp/WebConfig.java` - musim semi

```
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");
}
```

Mengonfigurasi strategi penamaan segmen

AWS X-Ray menggunakan nama layanan untuk mengidentifikasi aplikasi Anda dan membedakannya dari aplikasi lain, database, eksternal APIs, dan AWS sumber daya yang digunakan aplikasi Anda. Saat SDK X-Ray membuat segmen untuk permintaan masuk, SDK akan mencatat nama layanan aplikasi Anda di [kolom nama](#).

SDK X-Ray dapat memberi nama segmen setelah nama host di header permintaan HTTP. Namun, header ini dapat ditiru, yang dapat mengakibatkan simpul tak terduga di peta layanan Anda. Untuk mencegah SDK dari penamaan segmen salah karena permintaan dengan header host palsu, Anda harus menentukan nama default untuk permintaan masuk.

Jika aplikasi Anda menyuguhkan permintaan untuk beberapa domain, Anda dapat mengonfigurasi SDK untuk menggunakan strategi penamaan dinamis untuk mencerminkan ini dalam nama segmen. Strategi penamaan dinamis mengizinkan SDK menggunakan nama host untuk permintaan yang sesuai dengan pola yang diharapkan, dan menerapkan nama default untuk permintaan yang tidak sesuai.

Misalnya, Anda boleh memiliki satu aplikasi yang melayani permintaan untuk tiga subdomain—`www.example.com`, `api.example.com`, dan `static.example.com`. Anda dapat menggunakan strategi penamaan dinamis dengan pola `*.example.com` untuk mengidentifikasi segmen untuk setiap subdomain dengan nama yang berbeda, mengakibatkan tiga simpul layanan pada peta layanan. Jika aplikasi Anda menerima permintaan dengan nama host yang tidak cocok dengan pola, Anda akan melihat simpul keempat pada peta layanan dengan nama fallback yang Anda tentukan.

Untuk menggunakan nama yang sama untuk semua segmen permintaan, tentukan nama aplikasi Anda ketika Anda memulai filter servlet, seperti yang ditampilkan dalam [bagian sebelumnya](#). Ini memiliki efek yang sama seperti membuat fixed [SegmentNamingStrategy](#) dengan memanggil `SegmentNamingStrategy.fixed()` dan meneruskannya ke [AWSXRayServletFilter](#) konstruktor.

Note

Anda dapat menimpa nama layanan default yang Anda tentukan dalam kode dengan [variabel lingkungan](#) `AWS_XRAY_TRACING_NAME`.

Strategi penamaan dinamis menentukan pola yang harus sesuai dengan nama host, dan nama default untuk digunakan jika nama host dalam permintaan HTTP tidak cocok dengan pola.

Untuk nama segmen dinamis di Tomcat, gunakan `dynamicNamingRecognizedHosts` dan `dynamicNamingFallbackName` untuk menentukan pola dan nama default, masing-masing.

Example WEB-INF/web.xml - filter servlet dengan penamaan dinamis

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.java.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>
```

Untuk Spring, buat dinamika [SegmentNamingStrategy](#) dengan memanggil `SegmentNamingStrategy.dynamic()`, dan meneruskannya ke `AWSXRayServletFilter` konstruktor.

Example `src/main/java/myapp/WebConfig.java` - filter servlet dengan penamaan dinamis

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.java.servlet.AWSXRayServletFilter;
```

```
import com.amazonaws.xray.strategy.SegmentNamingStrategy;

@Configuration
public class WebConfig {

    @Bean
    public Filter TracingFilter() {
        return new AWSXRayServletFilter(SegmentNamingStrategy.dynamic\("MyApp",
"\*.example.com"\));
    }
}
```

Menelusuri panggilan AWS SDK dengan X-Ray SDK for Java

Saat aplikasi Anda melakukan panggilan Layanan AWS untuk menyimpan data, menulis ke antrian, atau mengirim notifikasi, X-Ray SDK for Java melacak panggilan hilir di subsegmen. Ditelusuri Layanan AWS dan sumber daya yang Anda akses dalam layanan tersebut (misalnya, bucket Amazon S3 atau antrian Amazon SQS), muncul sebagai node hilir pada peta jejak di konsol X-Ray.

X-Ray SDK for Java secara otomatis menginstrumentasikan semua klien AWS SDK ketika Anda menyertakan [submodul](#) `aws-sdk` dan suatu `aws-sdk-instrumentor` dalam pembangunan Anda. Jika Anda tidak menyertakan submodul `Instrumentor`, Anda dapat memilih untuk menginstrumentasikan beberapa klien selagi mengeliminasi yang lain.

Untuk menginstrumentasikan klien individual, hapus `aws-sdk-instrumentor` submodule dari build Anda dan tambahkan `XRayClient` as a `TracingHandler` pada klien AWS SDK Anda menggunakan pembuat klien layanan.

Misalnya, untuk menginstrumentasikan klien `AmazonDynamoDB`, teruskan penanganan pelacakan ke `AmazonDynamoDBClientBuilder`.

Example `MyModel.java` - `DynamoDB` klien

```
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();  
...
```

Untuk semua layanan, Anda dapat melihat nama API yang dipanggil di konsol X-Ray. Untuk subset layanan, X-Ray SDK menambahkan informasi ke segmen untuk memberikan lebih banyak perincian di peta layanan.

Sebagai contoh, ketika Anda melakukan panggilan dengan klien DynamoDB berinstrumen, SDK menambahkan nama tabel ke segmen untuk panggilan yang menargetkan tabel. Di konsol tersebut, setiap tabel muncul sebagai simpul terpisah di peta layanan, dengan simpul DynamoDB generik untuk panggilan yang tidak menargetkan tabel.

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item

```
{  
  "id": "24756640c0d0978a",  
  "start_time": 1.480305974194E9,  
  "end_time": 1.4803059742E9,  
  "name": "DynamoDB",  
  "namespace": "aws",  
  "http": {  
    "response": {  
      "content_length": 60,  
      "status": 200  
    }  
  },  
  "aws": {  
    "table_name": "scorekeep-user",  
    "operation": "UpdateItem",  
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",  
  }  
}
```

Ketika Anda mengakses sumber daya bernama, panggilan ke layanan berikut membuat simpul tambahan di peta layanan. Panggilan yang tidak menargetkan sumber daya tertentu membuat simpul generik untuk layanan tersebut.

- Amazon DynamoDB – Nama tabel
- Amazon Simple Storage Service – Bucket dan nama kunci
- Amazon Simple Queue Service – Nama antrean

Untuk menginstrumentasikan panggilan hilir ke Layanan AWS AWS SDK untuk Java 2.2 dan yang lebih baru, Anda dapat menghilangkan `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` modul dari konfigurasi build Anda. Sebaliknya, sertakan `aws-xray-recorder-sdk-aws-sdk-v2` module, lalu instrumentasikan klien individu dengan mengonfigurasi mereka dengan `TracingInterceptor`.

Example AWS SDK untuk Java 2.2 dan kemudian - melacak pencegat

```
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();
    //...
```

Menelusuri panggilan ke layanan web downstream HTTP dengan X-Ray SDK for Java

Ketika aplikasi Anda membuat panggilan ke microservices atau HTTP publik APIs, Anda dapat menggunakan X-Ray SDK untuk versi Java `HttpClient` untuk instrumen panggilan tersebut dan menambahkan API ke grafik layanan sebagai layanan hilir.

X-Ray SDK for Java `DefaultHttpClient` mencakup `HttpClientBuilder` dan kelas yang dapat digunakan sebagai pengganti `HttpComponents Apache` yang setara dengan instrumen panggilan HTTP keluar.

- `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`

Pustaka ini berada di submodul [aws-xray-recorder-sdk-apache-http](#).

Anda dapat mengganti pernyataan impor yang ada dengan X-Ray yang setara untuk instrumen semua klien, atau menggunakan nama yang sepenuhnya memenuhi syarat saat Anda menginisialisasi klien untuk instrumen klien tertentu.

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();
    }
}
```

Ketika Anda menginstrumen panggilan ke api web downstream, X-Ray SDK for Java mencatat subsegmen dengan informasi tentang permintaan dan respons HTTP. X-Ray menggunakan subsegmen untuk membuat segmen disimpulkan untuk API jarak jauh.

Example Subsegmen untuk panggilan HTTP downstream

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
```

```

"http": {
  "request": {
    "method": "GET",
    "url": "https://names.example.com/"
  },
  "response": {
    "content_length": -1,
    "status": 200
  }
}
}

```

Example Segmen yang disimpulkan untuk panggilan HTTP downstream

```

{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}

```

Menelusuri kueri SQL dengan X-Ray SDK for Java

Pencegat SQL

Instrumen kueri basis data SQL dengan menambahkan X-Ray SDK for Java JDBC pencegat untuk konfigurasi sumber data Anda.

- PostgreSQL – `com.amazonaws.xray.sql.postgres.TracingInterceptor`

- MySQL – `com.amazonaws.xray.sql.mysql.TracingInterceptor`

Pencegat ini berada di [aws-xray-recorder-sql-postgres](#) dan submodul [aws-xray-recorder-sql-mysql](#), masing-masing. Mereka menerapkan `org.apache.tomcat.jdbc.pool.JdbcInterceptor` dan kompatibel dengan koneksi kolam Tomcat.

Note

SQL pencegat tidak mencatat kueri SQL itu sendiri dalam subsegmen untuk tujuan keamanan.

Untuk Spring, tambahkan pencegat dalam file properti dan membangun sumber data dengan Spring Boot `DataSourceBuilder`.

Example **`src/main/java/resources/application.properties`** - PostgreSQL JDBC pencegat

```
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`** - Sumber data

```
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();
    }
    ...
}

```

Untuk Tomcat, panggilan `setJdbcInterceptors` pada sumber data JDBC dengan referensi ke kelas X-Ray SDK for Java.

Example **src/main/myapp/model.java** - Sumber data

```

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");

```

Pustaka Sumber Data JDBC Tomcat disertakan dalam X-Ray SDK for Java, tetapi Anda dapat mendeklarasikannya sebagai dependensi yang disediakan untuk dokumen yang Anda gunakan.

Example **pom.xml** - Sumber data JDBC.

```

<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jdbc</artifactId>
  <version>8.0.36</version>
  <scope>provided</scope>

```

```
</dependency>
```

Dekorator Penelusuran SQL Asli

- Tambahkan [aws-xray-recorder-sdk-sql](#) ke dependensi Anda.
- Hiasi sumber data database, koneksi, atau pernyataan Anda.

```
dataSource = TracingDataSource.decorate(dataSource)
connection = TracingConnection.decorate(connection)
statement = TracingStatement.decorateStatement(statement)
preparedStatement = TracingStatement.decoratePreparedStatement(preparedStatement,
    sql)
callableStatement = TracingStatement.decorateCallableStatement(callableStatement,
    sql)
```

Membuat subsegmen kustom dengan X-Ray SDK for Java

Subsegmen memperpanjang [segmen](#) penelusuran dengan detail tentang pekerjaan yang dilakukan untuk melayani permintaan. Setiap kali Anda melakukan panggilan dengan klien berinstrumen, X-Ray tersebut mencatat informasi yang dihasilkan dalam subsegmen. Anda dapat membuat subsegment tambahan untuk mengelompokkan subsegment lain, untuk mengukur performa bagian kode, atau untuk mencatat anotasi dan metadata.

Untuk mengelola subsegment, gunakan metode `beginSubsegment` dan `endSubsegment`.

Example `GameModel.java` - subsegmen kustom

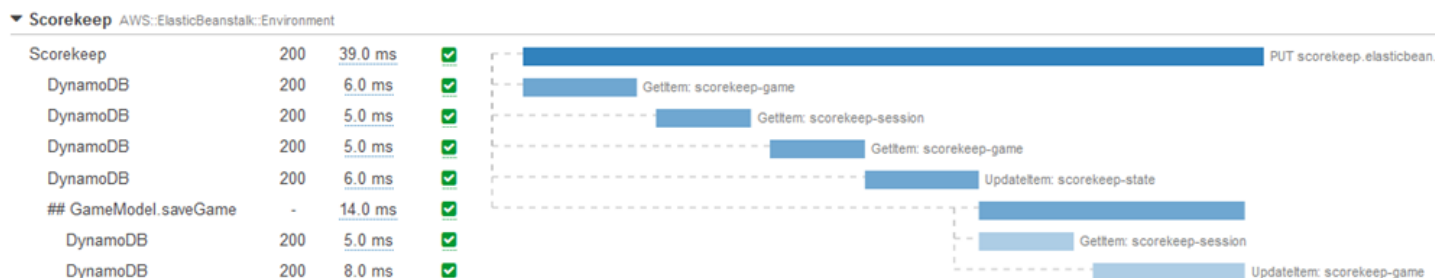
```
import com.amazonaws.xray.AWSXRay;
...
public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("Save Game");
    try {
        // check session
        String sessionId = game.getSession();
        if (sessionModel.loadSession(sessionId) == null ) {
            throw new SessionNotFoundException(sessionId);
        }
        mapper.save(game);
    } catch (Exception e) {
```

```

    subsegment.addException(e);
    throw e;
} finally {
    AWSXRay.endSubsegment();
}
}

```

Dalam contoh ini, kode dalam subsegmen memuat sesi game dari DynamoDB dengan metode pada model sesi, dan menggunakan AWS SDK untuk Java pemetaan DynamoDB untuk menyimpan game. Membungkus kode ini di subsegmen membuat panggilan DynamoDB anak subsegmen Save Game dalam tampilan penelusuran di konsol tersebut.



Jika kode di subsegmen Anda melempar pengecualian yang diperiksa, bungkus dalam blok `try` dan panggilan `AWSXRay.endSubsegment()` dalam blok `finally` untuk memastikan bahwa subsegmen selalu tertutup. Jika subsegmen tidak ditutup, segmen induk tidak dapat diselesaikan dan tidak akan dikirim ke X-Ray.

Untuk kode yang tidak membuang pengecualian yang diperiksa, Anda dapat melewati kode untuk `AWSXRay.CreateSubsegment` sebagai fungsi Lambda.

Example Subsegmen fungsi Lambda

```

import com.amazonaws.xray.AWSXRay;

AWSXRay.createSubsegment("getMovies", (subsegment) -> {
    // function code
});

```

Ketika Anda membuat subsegmen dalam segmen atau subsegmen lain, X-Ray SDK for Java membuat ID untuknya dan mencatat waktu mulai dan waktu akhir.

Example Subsegmen dengan metadata

```

"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"
  }
},

```

Untuk pemrograman asinkron dan multi-threaded, Anda harus secara manual melewati subsegment ke metode `endSubsegment()` untuk memastikan ditutup dengan benar karena konteks X-Ray dapat diubah selama eksekusi asinkron. Jika subsegmen asinkron ditutup setelah segmen induknya ditutup, metode ini akan secara otomatis mengalirkan seluruh segmen ke daemon X-Ray.

Example Subsegmen Asinkron

```

@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();
}

```

Tambahkan anotasi dan metadata ke segmen dengan X-Ray SDK for Java

Anda dapat mencatat informasi tambahan tentang permintaan, lingkungan, atau aplikasi dengan anotasi dan metadata. Anda dapat menambahkan anotasi dan metadata ke segmen yang dibuat oleh SDK X-Ray, atau subsegmen kustom yang Anda buat.

Anotasi adalah pasangan kunci-nilai dengan string, nomor, atau nilai-nilai Boolean. Anotasi diindekskan untuk digunakan dengan [Ekspresi filter](#). Gunakan anotasi untuk mencatat data yang

ingin Anda gunakan untuk mengelompokkan pelacakan di konsol tersebut, atau saat memanggil API [GetTraceSummaries](#).

Metadata adalah pasangan kunci-nilai yang dapat memiliki nilai dari setiap tipe, termasuk objek dan daftar, tetapi tidak diindekskan untuk digunakan dengan ekspresi filter. Gunakan metadata untuk mencatat data tambahan yang ingin disimpan dalam pelacakan tetapi tidak perlu digunakan dengan pencarian.

Selain anotasi dan metadata, Anda juga dapat [mencatat string ID pengguna](#) pada segmen. Pengguna IDs direkam dalam bidang terpisah pada segmen dan diindeks untuk digunakan dengan pencarian.

Bagian-bagian

- [Mencatat anotasi dengan X-Ray SDK for Java](#)
- [Mencatat metadata dengan X-Ray SDK for Java](#)
- [Merekam pengguna IDs dengan X-Ray SDK for Java](#)

Mencatat anotasi dengan X-Ray SDK for Java

Gunakan anotasi untuk mencatat informasi pada segmen atau subsegmen yang ingin diindeks untuk pencarian.

Persyaratan Anotasi

- Tombol — Kunci untuk anotasi X-Ray dapat memiliki hingga 500 karakter alfanumerik. Anda tidak dapat menggunakan spasi atau simbol selain titik atau titik (.).
- Nilai — Nilai untuk anotasi X-Ray dapat memiliki hingga 1.000 karakter Unicode.
- Jumlah Anotasi — Anda dapat menggunakan hingga 50 anotasi per jejak.

Untuk mencatat anotasi

1. Dapatkan referensi ke segmen atau subsegmen saat ini dari AWSXRay.

```
import com.amazonaws.xray.AWSXRay;  
import com.amazonaws.xray.entities.Segment;  
...  
Segment document = AWSXRay.getCurrentSegment();
```

atau

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Subsegment;
...
Subsegment document = AWSXRay.getCurrentSubsegment();
```

2. Panggil `putAnnotation` dengan kunci String, serta nilai Boolean, Nomor, atau String.

```
document.putAnnotation("mykey", "my value");
```

Contoh berikut menunjukkan cara memanggil `putAnnotation` dengan kunci String yang mencakup titik, dan nilai Boolean, Number, atau String.

```
document.putAnnotation("testkey.test", "my value");
```

SDK mencatat penjelasan sebagai pasangan nilai kunci dalam objek annotations di dokumen segmen. Memanggil `putAnnotation` dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Untuk menemukan penelusuran yang memiliki anotasi dengan nilai-nilai tertentu, gunakan kata kunci `annotation[key]` dalam [ekspresi filter](#).

Example [src/main/java/scorekeep/GameModel.java](#) – Anotasi dan metadata

```
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();
}
}
```

Mencatat metadata dengan X-Ray SDK for Java

Gunakan metadata untuk mencatat informasi pada segmen atau subsegmen yang tidak perlu diindeks untuk pencarian. Nilai metadata bisa string, angka, Boolean, atau objek yang dapat diserialkan ke dalam objek JSON atau array.

Untuk mencatat metadata

1. Dapatkan referensi ke segmen atau subsegmen saat ini dari AWSXRay.

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
...
Segment document = AWSXRay.getCurrentSegment();
```

atau

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Subsegment;
...
Subsegment document = AWSXRay.getCurrentSubsegment();
```

2. Panggil `putMetadata` dengan namespace String, kunci String, dan nilai Boolean, Nomor, String, atau Objek.

```
document.putMetadata("my namespace", "my key", "my value");
```

atau

Panggil `putMetadata` hanya dengan kunci dan nilai.

```
document.putMetadata("my key", "my value");
```

Jika Anda tidak menentukan namespace, SDK menggunakan default. Memanggil `putMetadata` dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Example [src/main/java/scorekeep/GameModel.java](#) – Anotasi dan metadata

```
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();
    }
}
```

Merekam pengguna IDs dengan X-Ray SDK for Java

Rekam segmen pengguna IDs berdasarkan permintaan untuk mengidentifikasi pengguna yang mengirim permintaan.

Untuk merekam pengguna IDs

1. Dapatkan referensi ke segmen saat ini dari AWSXRay.

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
...
Segment document = AWSXRay.getCurrentSegment();
```

2. Panggil setUser dengan ID String pengguna yang mengirim permintaan.

```
document.setUser("U12345");
```

Anda dapat menelepon setUser di pengendali Anda untuk mencatat ID pengguna segera setelah aplikasi mulai memproses permintaan. Jika Anda hanya akan menggunakan segmen untuk mengatur ID pengguna, Anda dapat mengaitkan panggilan dalam satu baris.

Example [src/main/java/scorekeep/MoveController.java](#) — ID Pengguna

```
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);
}
```

Untuk menemukan pelacakan untuk ID pengguna, gunakan kata kunci user dalam [Ekspresi filter](#).

AWS X-Ray metrik untuk X-Ray SDK for Java

Topik ini menjelaskan AWS X-Ray namespace, metrik, dan dimensi. Anda dapat menggunakan X-Ray SDK for Java untuk mempublikasikan metrik CloudWatch Amazon tanpa sampel dari segmen X-Ray yang dikumpulkan. Metrik ini berasal dari waktu mulai dan akhir segmen, dan kesalahan, dan bendera status yang dibatasi. Gunakan metrik pelacakan ini untuk menampilkan masalah pengulangan dan ketergantungan dalam subsegmen.

CloudWatch adalah repositori metrik. Metrik adalah konsep dasar dalam CloudWatch dan mewakili serangkaian titik data yang diatur waktu. Anda (atau Layanan AWS) mempublikasikan titik data metrik ke dalam CloudWatch dan Anda mengambil statistik tentang titik data tersebut sebagai kumpulan data deret waktu yang diurutkan.

Metrik ditentukan secara unik dari suatu nama, namespace, dan satu dimensi atau lebih. Setiap titik data dalam metrik memiliki stempel waktu, dan secara opsional, unit pengukuran. Bila Anda meminta statistik, aliran data yang dikembalikan akan diidentifikasi dengan namespace, nama metrik dan dimensi.

Untuk informasi selengkapnya CloudWatch, lihat [Panduan CloudWatch Pengguna Amazon](#).

CloudWatch Metrik X-Ray

Namespace `ServiceMetrics/SDK` mencakup metrik berikut.

Metrik	Statistik tersedia	Deskripsi	Unit
Latency	Rata-rata, Minimum, Maksimum, Total	Perbedaan antara waktu mulai dan akhir. Rata-rata, minimum, dan maksimum, semuanya menggambarkan latensi operasional. Total menggambarkan jumlah panggilan.	Milidetik
ErrorRate	Rata-rata, Jumlah	Tingkat permintaan yang gagal dengan kode status 4xx Client Error, mengakibatkan kesalahan.	Persen
FaultRate	Rata-rata, Jumlah	Tingkat penelusuran yang gagal dengan kode status 5xx	Persen

Metrik	Statistik tersedia	Deskripsi	Unit
		Server Error, mengakibatkan kesalahan.	
ThrottleRate	Rata-rata, Jumlah	Tingkat penelusuran yang dihentikan yang mengembalikan kode status 429. Ini adalah subset metrik <code>ErrorRate</code> .	Persen
OkRate	Rata-rata, Jumlah	Tingkat permintaan yang ditelusuri menghasilkan kode status OK.	Persen

CloudWatch Dimensi X-Ray

Gunakan dimensi dalam tabel berikut untuk menyempurnakan metrik yang dikembalikan untuk instrumen X-Ray Anda Java aplikasi.

Dimensi	Deskripsi
ServiceType	Tipe layanan, misalnya <code>AWS::EC2::Instance</code> atau <code>NONE</code> , jika tidak diketahui.
ServiceName	Nama kanonis untuk layanan ini.

Aktifkan CloudWatch metrik X-Ray

Gunakan prosedur berikut untuk mengaktifkan metrik penelusuran di instrumen Anda Java aplikasi.

Untuk mengonfigurasi metrik penelusuran

1. Tambahkan `aws-xray-recorder-sdk-metrics` paket sebagai Apache Maven ketergantungan. Untuk informasi selengkapnya, lihat [Submodul X-Ray SDK for Java](#).

2. Aktifkan `MetricsSegmentListener()` baru sebagai bagian dari pembangunan pencatatan global

Example `src/com/myapp/web/Startup.java`

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.plugins.ElasticBeanstalkPlugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

@Configuration
public class WebConfig {
    ...
    static {
        AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
            .standard()
            .withPlugin(new EC2Plugin())
            .withPlugin(new ElasticBeanstalkPlugin())
            .withSegmentListener(new
MetricsSegmentListener());

        URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
        builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

        AWSXRay.setGlobalRecorder(builder.build());
    }
}
```

3. Terapkan CloudWatch agen untuk mengumpulkan metrik menggunakan Amazon Elastic Compute Cloud (Amazon) EC2, Amazon Elastic Container Service (Amazon ECS) Service Elastic Container (Amazon ECS), atau Amazon Elastic Kubernetes Service (Amazon EKS):
 - Untuk mengonfigurasi Amazon EC2, lihat [Menginstal CloudWatch agen](#).
 - Untuk mengonfigurasi Amazon ECS, lihat [Memantau kontainer Amazon ECS menggunakan Wawasan Kontainer](#).
 - Untuk mengonfigurasi Amazon EKS, lihat [Menginstal CloudWatch agen menggunakan add-on Amazon CloudWatch Observability EKS](#).

4. Konfigurasi SDK untuk berkomunikasi dengan CloudWatch agen. Secara default, SDK berkomunikasi dengan CloudWatch agen pada alamat. `127.0.0.1` Anda dapat mengonfigurasi alamat alternatif dengan menetapkan variabel lingkungan atau properti Java ke `address:port`.

Example Variabel Lingkungan

```
AWS_XRAY_METRICS_DAEMON_ADDRESS=address:port
```

Example Properti Java

```
com.amazonaws.xray.metrics.daemonAddress=address:port
```

Untuk memvalidasi konfigurasi

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Buka tab Metrik untuk mengamati masuknya metrik Anda.
3. (Opsional) Di CloudWatch konsol, pada tab Log, buka grup ServiceMetricsSDK log. Cari pengaliran log yang cocok dengan metrik host, dan konfirmasi pesan log.

Melewati konteks segmen antara benang dalam aplikasi multithreaded

Ketika Anda membuat thread baru dalam aplikasi Anda, `AWSXRayRecorder` tidak mempertahankan referensi ke segmen saat ini atau subsegmen [Entitas](#). Jika Anda menggunakan klien berinstrumen di utas baru, SDK akan mencoba menulis ke segmen yang tidak ada, menyebabkan file.

[SegmentNotFoundException](#)

Untuk menghindari melempar pengecualian selama pengembangan, Anda dapat mengonfigurasi perekam dengan [ContextMissingStrategy](#) yang memberitahunya untuk mencatat kesalahan sebagai gantinya. Anda dapat mengonfigurasi strategi dalam kode dengan [SetContextMissingStrategy](#), atau mengonfigurasi opsi yang setara dengan [variabel lingkungan](#) atau [properti sistem](#).

Salah satu cara untuk mengatasi kesalahan adalah dengan menggunakan segmen baru dengan memanggil [beginSegment](#) ketika Anda memulai thread dan [endSegment](#) ketika Anda menutupnya. Hal ini bekerja jika Anda menginstrumen kode yang tidak berjalan dalam menanggapi permintaan HTTP, seperti kode yang berjalan ketika aplikasi Anda dimulai.

Jika Anda menggunakan beberapa thread untuk menangani permintaan masuk, Anda dapat melewati segmen saat ini atau subsegmen untuk thread baru dan memberikan ke pencatat global. Hal ini memastikan bahwa informasi yang tercatat dalam thread baru dikaitkan dengan segmen yang sama dengan sisa informasi yang tercatat tentang permintaan tersebut. Setelah segmen tersedia di thread baru, Anda dapat mengeksekusi runnable apa pun dengan akses ke konteks segmen itu menggunakan metode `segment.run(() -> { ... })`.

Lihat [Menggunakan klien berinstrumen di utas pekerja](#) sebagai contoh.

Menggunakan X-Ray dengan Program Asinkron

X-Ray SDK for Java dapat digunakan dalam program Java asinkron dengan.

[SegmentContextExecutors](#) `SegmentContextExecutor` Mengimplementasikan antarmuka `Executor`, yang berarti dapat diteruskan ke semua operasi asinkron dari file. [CompletableFuture](#) Hal ini memastikan bahwa setiap operasi asinkron akan dieksekusi dengan segmen yang benar dalam konteksnya.

Example Contoh App.java: Meneruskan `SegmentContextExecutor` ke `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 dengan Spring dan X-Ray SDK for Java

Topik ini menjelaskan cara menggunakan SDK X-Ray dan Kerangka Kerja Spring untuk menginstrumentasikan aplikasi Anda tanpa mengubah logika intinya. Ini berarti bahwa sekarang ada cara non-invasif untuk instrumen aplikasi Anda berjalan dari jarak jauh. AWS

Untuk mengaktifkan AOP di Spring

1. [Konfigurasi Musim Semi](#)

2. [Tambahkan filter penelusuran ke aplikasi Anda](#)
3. [Beri anotasi kode Anda atau terapkan antarmuka](#)
4. [Aktifkan X-Ray di aplikasi Anda](#)

Mengonfigurasi Spring

Anda dapat menggunakan Maven atau Gradle untuk mengonfigurasi Spring menggunakan AOP untuk instrumen aplikasi Anda.

Jika Anda menggunakan Maven untuk membangun aplikasi Anda, tambahkan ketergantungan berikut di file `pom.xml` Anda.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-xray-recorder-sdk-spring</artifactId>
  <version>2.11.0</version>
</dependency>
```

Untuk Gradle, tambahkan ketergantungan berikut di file `build.gradle` Anda.

```
compile 'com.amazonaws:aws-xray-recorder-sdk-spring:2.11.0'
```

Mengkonfigurasi Spring Boot

Selain ketergantungan Spring yang dijelaskan di bagian sebelumnya, jika Anda menggunakan Spring Boot, tambahkan dependensi berikut jika belum ada di classpath Anda.

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'
```

Menambahkan filter pelacakan ke aplikasi Anda

Tambahkan `Filter` ke kelas `WebConfig` Anda. Teruskan nama segmen ke konstruktor [AWSXRayServletFilter](#) sebagai string. Untuk informasi selengkapnya tentang filter pelacakan dan menginstrumentasikan permintaan masuk, lihat [Menelusuri permintaan yang masuk dengan X-Ray SDK for Java](#).

Example `src/main/java/myapp/WebConfig.java` - musim semi

```
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");
    }
}
```

Support Jakarta

Spring 6 menggunakan [Jakarta](#) sebagai pengganti `javax` untuk Enterprise Edition. Untuk mendukung namespace baru ini, X-Ray telah menciptakan serangkaian kelas paralel yang hidup di namespace Jakarta mereka sendiri.

Untuk kelas filter, ganti `javax` dengan `jakarta`. Saat mengonfigurasi strategi penamaan segmen, tambahkan `jakarta` sebelum nama kelas strategi penamaan, seperti pada contoh berikut:

```
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"));  
    }  
}
```

Anotasi kode Anda atau menerapkan antarmuka

Kelas Anda harus dianotasikan dengan `@XRayEnabled`, atau menerapkan antarmuka `XRayTraced`. Langkah ini memberitahu sistem AOP untuk membungkus fungsi kelas yang terdampak untuk instrumentasi X-Ray.

Mengaktifkan X-Ray dalam aplikasi

Untuk mengaktifkan pelacakan X-Ray dalam aplikasi Anda, kode Anda harus memperpanjang kelas abstrak `BaseAbstractXRayInterceptor` dengan menimpa metode berikut.

- `generateMetadata`—Fungsi ini mengizinkan kustomisasi metadata yang terlampir pada pelacakan fungsi saat ini. Secara default, nama kelas dari fungsi yang sedang berjalan dicatat dalam metadata. Anda dapat menambahkan lebih banyak data jika Anda memerlukan informasi tambahan.
- `xrayEnabledClasses`—Fungsi ini kosong, dan harus tetap demikian. Berfungsi sebagai host untuk pointcut yang menginstruksikan penghalang tentang metode yang digunakan untuk membungkus. Tentukan pointcut dengan menentukan kelas yang dianotasikan dengan `@XRayEnabled` untuk melacak. Pernyataan pointcut berikut memberitahu penghalang untuk membungkus semua pengendali bean yang teranotasi dengan anotasi `@XRayEnabled`.

```
@Pointcut("@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)")
```

Jika proyek Anda menggunakan Spring Data JPA, pertimbangkan untuk memperpanjang dari `AbstractXRayInterceptor` bukan `BaseAbstractXRayInterceptor`.

Contoh

Kode berikut memperluas kelas abstrak `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() {}
}
```

Kode berikut adalah kelas yang akan diinstrumentasi oleh X-Ray.

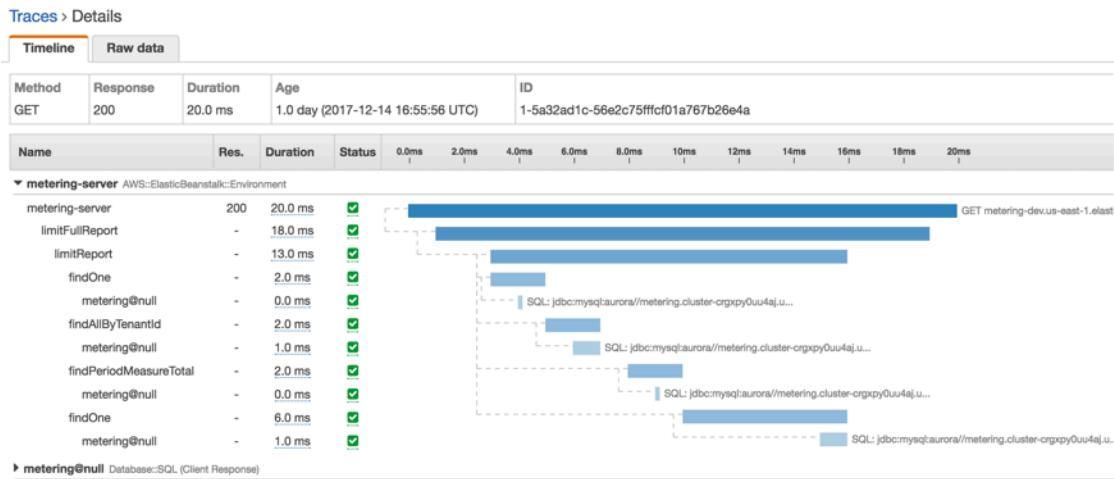
```
@Service
@XRayEnabled
public class MyServiceImpl implements MyService {
    private final MyEntityRepository myEntityRepository;

    @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());
        }
    }
}
```

Jika Anda telah mengonfigurasi aplikasi Anda dengan benar, Anda akan melihat tumpukan panggilan aplikasi yang lengkap, dari pengendali hingga panggilan layanan seperti yang ditunjukkan pada tangkapan layar konsol tersebut.



Bekerja menggunakan Node.js

Ada dua cara untuk menginstruksikan aplikasi Node.js Anda untuk mengirim jejak ke X-Ray:

- [AWS Distro untuk OpenTelemetry JavaScript - AWS Distribusi yang menyediakan serangkaian pustaka open source untuk mengirim metrik dan jejak yang berkorelasi ke beberapa solusi AWS pemantauan, termasuk Amazon, AWS X-Ray dan Amazon OpenSearch Service CloudWatch, melalui Distro for Collector.AWS OpenTelemetry](#)
- [AWS X-Ray SDK untuk Node.js - Satu set perpustakaan untuk menghasilkan dan mengirim jejak ke X-Ray melalui daemon X-Ray.](#)

Untuk informasi selengkapnya, lihat [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#).

AWS Distro untuk OpenTelemetry JavaScript

Dengan AWS Distro for OpenTelemetry (ADOT) JavaScript, Anda dapat menginstruksikan aplikasi Anda sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan termasuk Amazon CloudWatch, AWS X-Ray dan Amazon Service. OpenSearch Menggunakan X-Ray dengan AWS Distro untuk OpenTelemetry membutuhkan dua komponen: OpenTelemetry SDK diaktifkan untuk digunakan dengan X-Ray, dan AWS Distro untuk OpenTelemetry Kolektor diaktifkan untuk digunakan dengan X-Ray.

Untuk memulai, lihat [AWS Distro untuk OpenTelemetry JavaScript dokumentasi](#).

Note

ADOT JavaScript didukung untuk semua aplikasi Node.js sisi server. ADOT JavaScript tidak dapat mengeksport data ke X-Ray dari klien browser.

Untuk informasi selengkapnya tentang menggunakan AWS Distro untuk OpenTelemetry with AWS X-Ray dan lainnya Layanan AWS, lihat [AWS Distro untuk OpenTelemetry atau Distro untuk AWS Dokumentasi](#). OpenTelemetry

Untuk informasi selengkapnya tentang dukungan dan penggunaan bahasa, lihat [AWS Observabilitas di GitHub](#).

AWS X-Ray SDK untuk Node.js

X-Ray SDK untuk Node.js adalah pustaka untuk aplikasi web Express dan fungsi Node.js Lambda yang menyediakan kelas dan metode untuk menghasilkan dan mengirim pelacakan data ke daemon X-Ray. Data pelacakan mencakup informasi tentang permintaan HTTP masuk yang disajikan oleh aplikasi, dan panggilan yang dilakukan aplikasi ke layanan hilir menggunakan klien AWS SDK atau HTTP.

Note

X-Ray SDK untuk Node.js adalah proyek open source yang didukung untuk Node.js versi 14.x dan lebih tinggi. Anda dapat mengikuti proyek dan mengirimkan masalah dan menarik permintaan di GitHub: github.com/aws/aws-xray-sdk-node

Jika Anda menggunakan Express, mulai dengan [menambahkan SDK sebagai middleware](#) pada server aplikasi Anda untuk pelacakan permintaan masuk. Middleware membuat [segmen](#) untuk setiap permintaan yang dilacak, dan menyelesaikan segmen ketika tanggapan dikirim. Ketika segmen terbuka Anda dapat menggunakan metode klien SDK untuk menambahkan informasi ke segmen dan membuat subsegmen untuk pelacakan panggilan hilir. SDK juga secara otomatis mencatat pengecualian yang aplikasi Anda lempar ketika segmen terbuka.

Untuk fungsi Lambda disebut oleh instrumen aplikasi atau layanan, Lambda membaca [tracing header](#) dan pelacakan sampel permintaan secara otomatis. Untuk fungsi lainnya, Anda dapat [mengonfigurasi Lambda](#) untuk sampel dan pelacakan permintaan masuk. Dalam kedua kasus, Lambda membuat segmen dan menyediakannya ke X-Ray SDK.

Note

Pada Lambda, X-Ray SDK adalah opsional. Jika Anda tidak menggunakannya dalam fungsi Anda, peta layanan Anda masih akan menyertakan simpul untuk layanan Lambda, dan satu untuk setiap fungsi Lambda. Dengan menambahkan SDK, Anda dapat melakukan instrumen kode fungsi Anda untuk menambahkan subsegmen ke segmen fungsi yang dicatat oleh Lambda. Lihat [AWS Lambda dan AWS X-Ray](#) untuk informasi selengkapnya.

Selanjutnya, gunakan X-Ray SDK untuk Node.js untuk [instrumen AWS SDK Anda JavaScript di klien Node.js](#). Setiap kali Anda melakukan panggilan ke hilir Layanan AWS atau sumber daya dengan klien

yang diinstrumentasi, SDK akan mencatat informasi tentang panggilan di subsegmen. Layanan AWS dan sumber daya yang Anda akses dalam layanan muncul sebagai node hilir pada peta jejak untuk membantu Anda mengidentifikasi kesalahan dan masalah pembatasan pada koneksi individual.

X-Ray SDK untuk Node.js juga menyediakan instrumentasi untuk panggilan hilir ke web HTTP APIs dan kueri SQL. [Bungkus klien HTTP Anda dalam metode penangkapan SDK](#) untuk mencatat informasi tentang panggilan HTTP keluar. Untuk klien SQL, [menggunakan metode penangkapan untuk tipe basis data Anda](#).

Middleware menerapkan aturan pengambilan sampel ke permintaan masuk untuk menentukan permintaan yang dilacak. Anda dapat [mengonfigurasi X-Ray SDK untuk Node.js](#) untuk menyesuaikan perilaku pengambilan sampel atau merekam informasi tentang sumber daya AWS komputasi tempat aplikasi Anda berjalan.

Catat informasi tambahan tentang permintaan dan pekerjaan yang dilakukan aplikasi Anda dalam [anotasi dan metadata](#). Anotasi adalah pasangan kunci-nilai sederhana yang diindeks untuk digunakan dengan [ekspresi filter](#), sehingga Anda dapat mencari pelacakan yang berisi data tertentu. Entri metadata kurang bersifat membatasi dan dapat mencatat seluruh objek dan array — segala yang dapat disambungkan ke dalam JSON.

Anotasi dan Metadata

Anotasi dan metadata adalah teks abritari yang Anda tambahkan ke segmen dengan X-Ray SDK. Anotasi diindekskan untuk digunakan dengan ekspresi filter. Metadata tidak diindeks, tetapi dapat dilihat di segmen mentah dengan konsol X-Ray atau API. Siapa pun yang Anda berikan akses baca ke X-Ray dapat melihat data ini.

Bila Anda memiliki banyak klien diinstrumentasi dalam kode Anda, segmen permintaan tunggal dapat berisi sejumlah besar subsegmen, satu untuk setiap panggilan yang dilakukan dengan klien yang diinstrumentasi. Anda dapat mengatur dan mengelompokkan subsegmen dengan menggabungkan panggilan klien di [subsegmen kustom](#). Anda dapat membuat subsegmen kustom untuk seluruh fungsi atau bagian dari kode, dan mencatat metadata dan anotasi pada subsegmen bukan menulis segala sesuatu pada segmen induk.

Untuk dokumentasi referensi tentang kelas SDK dan metode, lihat [AWS X-Ray Referensi API SDK untuk Node.js](#).

Persyaratan

X-Ray SDK untuk Node.js membutuhkan Node.js dan pustaka berikut:

- `atomic-batcher` – 1.0.2
- `cls-hooked` – 4.2.2
- `pkginfo` – 0.4.0
- `semver` – 5.3.0

SDK menarik pustaka ini saat Anda menginstalnya dengan NPM.

Untuk melacak klien AWS SDK, X-Ray SDK untuk Node.js memerlukan versi minimum AWS SDK untuk JavaScript Node.js.

- `aws-sdk` – 2.7.15

Manajemen dependensi

X-Ray SDK untuk Node.js tersedia dari NPM.

- Package - [aws-xray-sdk](#)

Untuk pengembangan lokal, instal SDK di direktori proyek Anda dengan 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
```

Gunakan opsi `--save` untuk menyimpan SDK sebagai dependensi dalam aplikasi `package.json` Anda.

```
~/nodejs-xray$ npm install aws-xray-sdk --save
aws-xray-sdk@3.3.3
```

Jika aplikasi Anda memiliki dependensi yang versinya bertentangan dengan dependensi X-Ray SDK, kedua versi akan diinstal untuk memastikan kompatibilitas. Untuk detail selengkapnya, lihat [dokumentasi NPM resmi untuk resolusi dependensi](#).

Sampel Node.js

Bekerja dengan AWS X-Ray SDK untuk Node.js untuk mendapatkan end-to-end tampilan permintaan saat mereka melakukan perjalanan melalui aplikasi Node.js Anda.

- [Node.js contoh aplikasi](#) pada GitHub.

Mengonfigurasi SDK X-Ray for Node.js

Anda dapat mengonfigurasi SDK X-Ray for Node.js dengan plugin untuk menyertakan informasi tentang layanan yang dijalankan aplikasi Anda, mengubah perilaku pengambilan sampel default, atau menambahkan aturan pengambilan sampel yang berlaku untuk permintaan ke jalur tertentu.

Bagian-bagian

- [Plugin layanan](#)
- [Aturan pengambilan sampel](#)
- [Pencatatan log](#)
- [Alamat daemon X-Ray](#)
- [Variabel-variabel lingkungan](#)

Plugin layanan

Gunakan `plugins` untuk mencatat informasi tentang layanan yang meng-hosting aplikasi Anda.

Plugin

- Amazon EC2 — EC2Plugin menambahkan ID instans, Availability Zone, dan Grup CloudWatch Log.
- Elastic Beanstalk – ElasticBeanstalkPlugin menambahkan nama lingkungan, label versi, dan ID deployment.
- Amazon ECS – ECSPlugin menambahkan ID kontainer.

Untuk menggunakan plugin, konfigurasikan klien X-Ray SDK for Node.js dengan menggunakan metode `config`.

Example `app.js` - plugin

```
var AWSXRay = require('aws-xray-sdk');  
AWSXRay.config([AWSXRay.plugins.EC2Plugin, AWSXRay.plugins.ElasticBeanstalkPlugin]);
```

SDK juga menggunakan pengaturan plugin untuk mengatur bidang `origin` pada segmen. Ini menunjukkan jenis AWS sumber daya yang menjalankan aplikasi Anda. Saat Anda menggunakan beberapa plugin, SDK menggunakan urutan resolusi berikut untuk menentukan asal: ElasticBeanstalk > EKS > ECS > EC2

Aturan pengambilan sampel

SDK menggunakan aturan pengambilan sampel yang Anda tetapkan di konsol X-Ray untuk menentukan permintaan yang akan dicatat. Aturan default menelusuri permintaan pertama setiap detik, dan lima persen permintaan tambahan di semua layanan yang mengirim pelacakan ke X-Ray. [Buat aturan tambahan di konsol X-Ray](#) untuk menyesuaikan jumlah data yang dicatat untuk setiap aplikasi Anda.

SDK menerapkan aturan kustom sesuai urutan penetapannya. Jika permintaan cocok dengan beberapa aturan kustom, SDK hanya menerapkan aturan pertama.

Note

Jika SDK tidak dapat mencapai X-Ray untuk mendapatkan aturan pengambilan sampel, SDK akan beralih ke aturan lokal default dari permintaan pertama setiap detik, dan lima persen permintaan tambahan per host. Hal ini dapat terjadi jika host tidak memiliki izin untuk

memanggil sampling APIs, atau tidak dapat terhubung ke daemon X-Ray, yang bertindak sebagai proxy TCP untuk panggilan API yang dilakukan oleh SDK.

Anda juga dapat mengonfigurasi SDK untuk memuat aturan sampling dari dokumen JSON. SDK dapat menggunakan aturan lokal sebagai cadangan jika terjadi kasus tidak dapat mengambil sampel X-Ray, atau menggunakan aturan lokal secara eksklusif.

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
  }
}
```

Contoh ini menentukan satu aturan kustom dan aturan default. Aturan kustom menerapkan tingkat pengambilan sampel lima persen tanpa jumlah minimum permintaan untuk melacak jalur di `/api/move/`. Aturan default menelusuri permintaan pertama setiap detik dan 10 persen dari permintaan tambahan.

Kerugian dari menentukan aturan secara lokal adalah bahwa target tetap diterapkan oleh setiap instans pencatat secara independen, alih-alih dikelola oleh layanan X-Ray. Ketika Anda men-deploy lebih banyak host, laju tetap akan dikalikan, sehingga sulit untuk mengontrol jumlah data yang dicatat.

Pada AWS Lambda, Anda tidak dapat mengubah laju pengambilan sampel. Jika fungsi Anda dipanggil oleh layanan yang diinstrumentasikan, panggilan yang menghasilkan permintaan yang

sampelnya diambil oleh layanan yang akan dicatat oleh Lambda. Jika pelacakan aktif diaktifkan dan tidak ada header pelacakan, Lambda membuat keputusan pengambilan sampel.

Untuk mengonfigurasi aturan cadangan, beri tahu SDK X-Ray for Node.js untuk memuat aturan pengambilan sampel dari file dengan `setSamplingRules`.

Example app.js - aturan pengambilan sampel dari suatu file

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
```

Anda juga dapat menentukan aturan dalam kode dan meneruskannya ke `setSamplingRules` sebagai objek.

Example app.js - aturan pengambilan sampel untuk suatu objek

```
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);
```

Untuk hanya menggunakan aturan lokal, hubungi `disableCentralizedSampling`.

```
AWSXRay.middleware.disableCentralizedSampling()
```

Pencatatan log

Untuk mencatat output dari SDK, panggil `AWSXRay.setLogger(logger)`, tempat `logger` merupakan obyek yang menyediakan metode pencatatan standar (`warn`, `info`, dll.).

Secara default SDK akan mencatat pesan kesalahan ke konsol tersebut menggunakan metode standar pada objek konsol. Tingkat log pencatat bawaan dapat diatur dengan menggunakan variabel lingkungan `AWS_XRAY_DEBUG_MODE` atau `AWS_XRAY_LOG_LEVEL`. Untuk daftar nilai tingkat log yang valid, lihat [Variabel lingkungan](#).

Jika Anda ingin memberikan format yang berbeda atau tujuan untuk log maka Anda dapat memberikan SDK dengan implementasi Anda sendiri dari antarmuka pencatat seperti yang ditunjukkan di bawah ini. Setiap objek yang mengimplementasikan antarmuka ini dapat digunakan. Yang berarti banyak pustaka pencatatan, misalnya Winston, dapat digunakan dan diteruskan ke SDK secara langsung.

Example app.js - pencatatan

```
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]);
```

Panggil `setLogger` sebelum Anda menjalankan metode konfigurasi lain untuk memastikan bahwa Anda menangkap output dari operasi tersebut.

Alamat daemon X-Ray

Jika daemon X-Ray mendengar di port atau host selain `127.0.0.1:2000`, Anda dapat mengonfigurasi X-Ray SDK for Node.js untuk mengirim data penelusuran ke alamat yang berbeda.

```
AWSXRay.setDaemonAddress('host:port');
```

Anda dapat menentukan host berdasarkan nama atau IPv4 alamat.

Example app.js - alamat daemon

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('daemonhost:8082');
```

Jika Anda mengonfigurasi daemon untuk mendengar di port yang berbeda untuk TCP dan UDP, Anda dapat menentukan keduanya dalam pengaturan alamat daemon.

Example app.js - alamat daemon pada port terpisah

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('tcp:daemonhost:8082 udp:daemonhost:8083');
```

Anda juga dapat mengatur alamat daemon dengan menggunakan [Variabel lingkungan](#) `AWS_XRAY_DAEMON_ADDRESS`.

Variabel-variabel lingkungan

Anda dapat menggunakan variabel lingkungan untuk mengonfigurasi X-Ray SDK for Node.js. SDK mendukung variabel berikut.

- `AWS_XRAY_CONTEXT_MISSING`— Setel `RUNTIME_ERROR` untuk melempar pengecualian saat kode instrumentasi Anda mencoba merekam data saat tidak ada segmen yang terbuka.

Nilai Valid

- `RUNTIME_ERROR`— Lempar pengecualian runtime.
- `LOG_ERROR`— Log kesalahan dan lanjutkan (default).
- `IGNORE_ERROR`— Abaikan kesalahan dan lanjutkan.

Kesalahan yang berkaitan dengan segmen atau subsegmen yang hilang dapat terjadi ketika Anda mencoba untuk menggunakan klien yang diinstrumentasi dalam kode perusahaan rintisan yang berjalan ketika tidak ada permintaan terbuka, atau dalam kode yang memunculkan thread baru.

- `AWS_XRAY_DAEMON_ADDRESS` – Mengatur host dan port pendengar daemon X-Ray. Secara default, SDK menggunakan `127.0.0.1:2000` untuk data pelacakan (UDP) dan pengambilan sampel (TCP). Gunakan variabel ini jika Anda telah mengonfigurasi daemon untuk [mendengarkan di port berbeda](#) atau jika berjalan pada host yang berbeda.

format

- Port yang sama – *address:port*
- Port yang berbeda – *tcp:address:port udp:address:port*
- `AWS_XRAY_DEBUG_MODE` – Atur ke `TRUE` untuk mengonfigurasi SDK untuk meng-output log ke konsol, di tingkat debug.
- `AWS_XRAY_LOG_LEVEL` – Mengatur tingkat log untuk pencatat default. Nilai yang valid adalah `debug`, `info`, `warn`, `error`, dan `silent`. Nilai ini diabaikan ketika `AWS_XRAY_DEBUG_MODE` disetel ke `TRUE`.

- `AWS_XRAY_TRACING_NAME` – Mengatur nama layanan yang digunakan SDK untuk segmen. Timpa nama segmen yang Anda [tetapkan di middleware Express](#).

Menelusuri permintaan yang masuk dengan X-Ray SDK for Node.js

Anda dapat menggunakan X-Ray SDK untuk Node.js untuk melacak permintaan HTTP masuk yang disajikan oleh aplikasi Express dan Restify pada instans EC2 di Amazon EC2, AWS Elastic Beanstalk atau Amazon ECS.

X-Ray SDK for Node.js menyediakan perangkat tengah untuk aplikasi yang menggunakan Express dan kerangka kerja Restify. Ketika Anda menambahkan perangkat tengah X-Ray ke aplikasi Anda, X-Ray SDK for Node.js membuat segmen untuk setiap permintaan sampel. Segmen ini mencakup waktu, metode, dan disposisi permintaan HTTP. Instrumentasi tambahan membuat subsegmen pada segmen ini.

Note

Untuk AWS Lambda fungsi, Lambda membuat segmen untuk setiap permintaan sampel. Untuk informasi selengkapnya, lihat [AWS Lambda dan AWS X-Ray](#).

Setiap segmen memiliki nama yang mengidentifikasi aplikasi Anda dalam peta layanan. Segmen dapat diberi nama secara statis, atau Anda dapat mengonfigurasi SDK untuk nama itu secara dinamis berdasarkan header host dalam permintaan masuk. Penamaan dinamis memungkinkan Anda mengelompokkan pelacakan berdasarkan nama domain dalam permintaan, dan menerapkan nama default jika nama tersebut tidak cocok dengan pola yang diharapkan (misalnya, jika header host ditiru).

Permintaan yang Diteruskan

Jika penyeimbang beban atau perantara lainnya meneruskan permintaan ke aplikasi Anda, X-Ray akan mengambil IP klien dari header `X-Forwarded-For` dalam permintaan bukan dari sumber IP dalam paket IP. IP klien yang dicatat untuk permintaan yang diteruskan dapat ditiru, sehingga tidak dapat dipercaya.

Ketika permintaan diteruskan, SDK menetapkan bidang tambahan di segmen untuk menunjukkan ini. Jika segmen yang berisi bidang `x_forwarded_for` ditetapkan ke `true`, IP klien diambil dari header `X-Forwarded-For` dalam permintaan HTTP.

Penangan pesan membuat segmen untuk setiap permintaan masuk dengan blok `http` yang berisi informasi berikut:

- Metode HTTP – DAPATKAN, POSTING, LETAKKAN, HAPUS, dll.
- Alamat klien – Alamat IP klien yang mengirim permintaan.
- Kode respons – Kode respons HTTP untuk permintaan yang selesai.
- Timing – Waktu mulai (saat permintaan diterima) dan waktu akhir (saat respons dikirim).
- Agen pengguna — `user-agent` dari permintaan.
- Panjang konten — `content-length` dari respons.

Bagian-bagian

- [Menelusuri permintaan masuk dengan Express](#)
- [Menelusuri permintaan masuk dengan restify](#)
- [Mengonfigurasi strategi penamaan segmen](#)

Menelusuri permintaan masuk dengan Express

Untuk menggunakan perangkat tengah Express, menginisialisasi klien SDK dan menggunakan perangkat tengah dikembalikan oleh fungsi `express.openSegment` sebelum Anda menentukan rute Anda.

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());
```

Setelah Anda menentukan rute Anda, gunakan output dari `express.closeSegment` seperti yang ditunjukkan untuk menangani kesalahan yang dikembalikan oleh X-Ray SDK for Node.js.

Menelusuri permintaan masuk dengan restify

Untuk menggunakan perangkat tengah Restify, menginisialisasi klien SDK dan menjalankan `enable`. Teruskan nama segmen dan server Restify Anda.

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');
});
```

Mengonfigurasi strategi penamaan segmen

AWS X-Ray menggunakan nama layanan untuk mengidentifikasi aplikasi Anda dan membedakannya dari aplikasi lain, database, eksternal APIs, dan AWS sumber daya yang digunakan aplikasi Anda. Saat SDK X-Ray membuat segmen untuk permintaan masuk, SDK akan mencatat nama layanan aplikasi Anda di [kolom nama](#).

SDK X-Ray dapat memberi nama segmen setelah nama host di header permintaan HTTP. Namun, header ini dapat ditiru, yang dapat mengakibatkan simpul tak terduga di peta layanan Anda. Untuk mencegah SDK dari penamaan segmen salah karena permintaan dengan header host palsu, Anda harus menentukan nama default untuk permintaan masuk.

Jika aplikasi Anda menyuguhkan permintaan untuk beberapa domain, Anda dapat mengonfigurasi SDK untuk menggunakan strategi penamaan dinamis untuk mencerminkan ini dalam nama segmen. Strategi penamaan dinamis mengizinkan SDK menggunakan nama host untuk permintaan yang sesuai dengan pola yang diharapkan, dan menerapkan nama default untuk permintaan yang tidak sesuai.

Misalnya, Anda boleh memiliki satu aplikasi yang melayani permintaan untuk tiga subdomain—`www.example.com`, `api.example.com`, dan `static.example.com`. Anda dapat menggunakan

strategi penamaan dinamis dengan pola `*.example.com` untuk mengidentifikasi segmen untuk setiap subdomain dengan nama yang berbeda, mengakibatkan tiga simpul layanan pada peta layanan. Jika aplikasi Anda menerima permintaan dengan nama host yang tidak cocok dengan pola, Anda akan melihat simpul keempat pada peta layanan dengan nama fallback yang Anda tentukan.

Untuk menggunakan nama yang sama untuk semua segmen permintaan, tentukan nama aplikasi Anda ketika Anda memulai perangkat tengah, seperti yang ditampilkan dalam bagian sebelumnya.

Note

Anda dapat menimpa nama layanan default yang Anda tentukan dalam kode dengan [variabel lingkungan](#) `AWS_XRAY_TRACING_NAME`.

Strategi penamaan dinamis menentukan pola yang harus sesuai dengan nama host, dan nama default untuk digunakan jika nama host dalam permintaan HTTP tidak cocok dengan pola. Untuk nama segmen secara dinamis, gunakan `AWSXRay.middleware.enableDynamicNaming`.

Example `app.js` - nama segmen dinamis

Jika nama host dalam permintaan cocok dengan pola `*.example.com`, gunakan nama host. Jika tidak sesuai, gunakan `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());
```

Menelusuri panggilan AWS SDK dengan X-Ray SDK untuk Node.js

[Saat aplikasi Anda melakukan panggilan Layanan AWS untuk menyimpan data, menulis ke antrian, atau mengirim notifikasi, X-Ray SDK untuk Node.js melacak panggilan hilir di subsegmen.](#) Ditelusuri

Layanan AWS, dan sumber daya yang Anda akses dalam layanan tersebut (misalnya, bucket Amazon S3 atau antrian Amazon SQS), muncul sebagai node hilir pada peta jejak di konsol X-Ray.

Klien AWS SDK instrumen yang Anda buat melalui [AWS SDK untuk JavaScript V2](#) atau [AWS SDK untuk JavaScript V3](#). Setiap versi AWS SDK menyediakan metode yang berbeda untuk menginstrumentasi klien AWS SDK.

Note

Saat ini, AWS X-Ray SDK untuk Node.js mengembalikan lebih sedikit informasi segmen saat menginstrumentasi klien AWS SDK untuk JavaScript V3, dibandingkan dengan menginstrumentasi klien V2. Misalnya, subsegmen yang mewakili panggilan ke DynamoDB tidak akan mengembalikan nama tabel. Jika Anda memerlukan informasi segmen ini di jejak Anda, pertimbangkan untuk menggunakan AWS SDK untuk JavaScript V2.

AWS SDK untuk JavaScript V2

Anda dapat AWS menginstruksikan semua klien SDK V2 dengan membungkus pernyataan `aws-sdk require` Anda dalam panggilan ke `AWSXRay.captureAWS`

Example `app.js` - AWS SDK instrumentasi

```
const AWS = AWSXRay.captureAWS(require('aws-sdk'));
```

Untuk menginstrumentasi klien individual, bungkus klien AWS SDK Anda dalam panggilan ke `AWSXRay.captureAWSCliient`. Misalnya, untuk instrumen klien `AmazonDynamoDB`:

Example `app.js` - instrumentasi klien `DynamoDB`

```
const AWSXRay = require('aws-xray-sdk');  
...  
const ddb = AWSXRay.captureAWSCliient(new AWS.DynamoDB());
```

Warning

Jangan gunakan kedua `captureAWS` dan `captureAWSCliient` bersama-sama. Hal ini akan menyebabkan duplikat subsegment.

Jika Anda ingin menggunakan [TypeScript](#) dengan [ECMAScript modul](#) (ESM) untuk memuat JavaScript kode, gunakan contoh berikut untuk mengimpor pustaka:

Example app.js - AWS Instrumentasi SDK

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

Untuk instrumen semua AWS klien dengan ESM, gunakan kode berikut:

Example app.js - AWS Instrumentasi 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();
```

Untuk semua layanan, Anda dapat melihat nama API yang dipanggil di konsol X-Ray. Untuk subset layanan, X-Ray SDK menambahkan informasi ke segmen untuk memberikan lebih banyak perincian di peta layanan.

Sebagai contoh, ketika Anda melakukan panggilan dengan klien DynamoDB berinstrumen, SDK menambahkan nama tabel ke segmen untuk panggilan yang menargetkan tabel. Di konsol tersebut, setiap tabel muncul sebagai simpul terpisah di peta layanan, dengan simpul DynamoDB generik untuk panggilan yang tidak menargetkan tabel.

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
```

```

    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}

```

Ketika Anda mengakses sumber daya bernama, panggilan ke layanan berikut membuat simpul tambahan di peta layanan. Panggilan yang tidak menargetkan sumber daya tertentu membuat simpul generik untuk layanan tersebut.

- Amazon DynamoDB – Nama tabel
- Amazon Simple Storage Service – Bucket dan nama kunci
- Amazon Simple Queue Service – Nama antrian

AWS SDK untuk JavaScript V3

AWS SDK untuk JavaScript V3 bersifat modular, jadi kode Anda hanya memuat modul yang dibutuhkannya. Karena itu, tidak mungkin untuk menginstruksikan semua klien AWS SDK karena V3 tidak mendukung metode `initCaptureAWS`.

Jika Anda ingin menggunakan TypeScript ECMAScript Modules (ESM) untuk memuat JavaScript kode Anda, Anda dapat menggunakan contoh berikut untuk mengimpor pustaka:

```

import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';

```

Instrumen setiap klien AWS SDK menggunakan `AWSXRay.captureAWSSv3Client` metode ini. Misalnya, untuk instrumen klien AmazonDynamoDB:

Example app.js - instrumentasi klien DynamoDB menggunakan SDK untuk Javascript V3

```

const AWSXRay = require('aws-xray-sdk');
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
...
const ddb = AWSXRay.captureAWSSv3Client(new DynamoDBClient({ region:
  "region" }));

```

Saat menggunakan AWS SDK untuk JavaScript V3, metadata seperti nama tabel, bucket dan nama kunci, atau nama antrian, saat ini tidak dikembalikan, dan oleh karena itu peta jejak tidak

akan berisi node diskrit untuk setiap sumber daya bernama seperti saat menginstrumentasi AWS klien SDK menggunakan V2. AWS SDK untuk JavaScript

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item, saat menggunakan V3 AWS SDK untuk JavaScript

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "operation": "UpdateItem",
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Menelusuri panggilan ke layanan web downstream HTTP menggunakan X-Ray SDK untuk Node.js

Saat aplikasi Anda melakukan panggilan ke microservices atau HTTP publik APIs, Anda dapat menggunakan klien X-Ray SDK for Node.js untuk instrumen panggilan tersebut dan menambahkan API ke grafik layanan sebagai layanan hilir.

Lewati http atau klien https untuk X-Ray SDK for Node.js metode `captureHTTPs` untuk menelusuri panggilan keluar.

Note

Panggilan menggunakan pustaka permintaan HTTP pihak ketiga, seperti Axios atau Superagent, didukung melalui [API `captureHTTPsGlobal\(\)`](#) dan masih akan ditelusuri ketika mereka menggunakan asli modul `http`.

Example app.js - klien HTTP

```
var AWSXRay = require('aws-xray-sdk');  
var http = AWSXRay.captureHTTPs(require('http'));
```

Untuk mengaktifkan penelusuran pada semua klien HTTP, panggilan `captureHTTPsGlobal` sebelum Anda memuat `http`.

Example app.js - klien HTTP (global)

```
var AWSXRay = require('aws-xray-sdk');  
AWSXRay.captureHTTPsGlobal(require('http'));  
var http = require('http');
```

Ketika Anda instrumen panggilan ke API web hilir, X-Ray SDK for Node.js mencatat subsegmen dengan informasi tentang permintaan HTTP dan respon. X-Ray menggunakan subsegmen untuk membuat segmen disimpulkan untuk API jarak jauh.

Example Subsegmen untuk panggilan HTTP downstream

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

Example Segmen yang disimpulkan untuk panggilan HTTP downstream

```
{
```

```

{id": "168416dc2ea97781",
"name": "names.example.com",
"trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
"start_time": 1484786387.131,
"end_time": 1484786387.501,
"parent_id": "004f72be19cddc2a",
"http": {
  "request": {
    "method": "GET",
    "url": "https://names.example.com/"
  },
  "response": {
    "content_length": -1,
    "status": 200
  }
},
"inferred": true
}

```

Menelusuri kueri SQL dengan X-Ray SDK for Node.js

Instrumen SQL kueri basis data dengan membungkus klien SQL Anda di X-Ray SDK yang sesuai untuk metode klien Node.js.

- PostgreSQL – `AWSXRay.capturePostgres()`

```

var AWSXRay = require('aws-xray-sdk');
var pg = AWSXRay.capturePostgres(require('pg'));
var client = new pg.Client();

```

- MySQL – `AWSXRay.captureMySQL()`

```

var AWSXRay = require('aws-xray-sdk');
var mysql = AWSXRay.captureMySQL(require('mysql'));
...
var connection = mysql.createConnection(config);

```

Ketika Anda menggunakan klien yang diinstrumentasi untuk membuat kueri SQL, X-Ray SDK untuk Node.js mencatat informasi tentang koneksi dan permintaan di subsegmen.

Termasuk data tambahan dalam SQL subsegmen

Anda dapat menambahkan informasi tambahan ke subsegmen yang dihasilkan untuk kueri SQL, asalkan dipetakan ke bidang SQL yang diizinkan. Misalnya, untuk mencatat string kueri SQL yang disterilkan di subsegmen, Anda dapat menambahkannya langsung ke subsegmen SQL objek.

Example Tetapkan SQL ke subsegmen

```
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;
}
```

Untuk daftar lengkap bidang SQL yang diizinkan, lihat [Kueri SQL](#) di Panduan Pengembang.AWS X-Ray

Menghasilkan subsegment kustom dengan X-Ray SDK untuk Node.js

Subsegmen memperluas [segmen](#) pelacakan dengan detail tentang pekerjaan yang dilakukan untuk melayani permintaan. Setiap kali Anda melakukan panggilan dengan klien berinstrumen, X-Ray tersebut mencatat informasi yang dihasilkan dalam subsegmen. Anda dapat membuat subsegment tambahan untuk mengelompokkan subsegment lain, untuk mengukur performa bagian kode, atau untuk mencatat anotasi dan metadata.

Subsegmen Express Kustom

Untuk membuat subsegmen kustom untuk fungsi yang membuat panggilan ke layanan hilir, gunakan fungsi `captureAsyncFunc`.

Example app.js - subsegment Express kustom

```
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();
};
```

Dalam contoh ini, aplikasi membuat subsegmen kustom bernama `send` untuk panggilan ke fungsi `sendRequest`. `captureAsyncFunc` melewati subsegmen yang harus Anda tutup dalam fungsi panggilan balik ketika panggilan asinkron yang dibuatnya selesai.

Untuk fungsi sinkron, Anda dapat menggunakan fungsi `captureFunc`, yang menutup subsegmen secara otomatis segera setelah blok fungsi selesai mengeksekusi.

Ketika Anda membuat subsegmen dalam segmen atau subsegmen lain, X-Ray SDK untuk Node.js menghasilkan ID untuk itu dan mencatat waktu mulai dan waktu berakhir.

Example Subsegmen dengan metadata

```
"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"
    }
  }
},
```

Subsegmen Lambda Kustom

SDK dikonfigurasi untuk secara otomatis membuat segmen fasad placeholder saat mendeteksi itu berjalan di Lambda. Untuk membuat subsegment dasar, yang akan membuat satu `AWS::Lambda::Function` node pada peta jejak X-Ray, panggil dan gunakan kembali segmen fasad. Jika Anda secara manual membuat segmen baru dengan ID baru (saat berbagi ID pelacakan, ID induk, dan sampel keputusan), Anda akan dapat mengirim segmen baru.

Example app.js - subsegment kustom manual

```
const segment = AWSXRay.getSegment(); //returns the facade segment
const subsegment = segment.addNewSubsegment('subseg');
...
subsegment.close();
//the segment is closed by the SDK automatically
```

Tambahkan anotasi dan metadata ke segmen dengan X-Ray SDK for Node.js

Anda dapat mencatat informasi tambahan tentang permintaan, lingkungan, atau aplikasi dengan anotasi dan metadata. Anda dapat menambahkan anotasi dan metadata ke segmen yang dibuat oleh SDK X-Ray, atau subsegmen kustom yang Anda buat.

Anotasi adalah pasangan kunci-nilai dengan string, nomor, atau nilai-nilai Boolean. Anotasi diindekskan untuk digunakan dengan [Ekspresi filter](#). Gunakan anotasi untuk mencatat data yang

ingin Anda gunakan untuk mengelompokkan pelacakan di konsol tersebut, atau saat memanggil API [GetTraceSummaries](#).

Metadata adalah pasangan kunci-nilai yang dapat memiliki nilai dari setiap tipe, termasuk objek dan daftar, tetapi tidak diindekskan untuk digunakan dengan ekspresi filter. Gunakan metadata untuk mencatat data tambahan yang ingin disimpan dalam pelacakan tetapi tidak perlu digunakan dengan pencarian.

Selain anotasi dan metadata, Anda juga dapat [mencatat string ID pengguna](#) pada segmen. Pengguna IDs direkam dalam bidang terpisah pada segmen dan diindeks untuk digunakan dengan pencarian.

Bagian-bagian

- [Mencatat anotasi dengan X-Ray SDK for Node.js](#)
- [Mencatat metadata dengan X-Ray SDK for Node.js](#)
- [Merekam pengguna IDs dengan X-Ray SDK untuk Node.js](#)

Mencatat anotasi dengan X-Ray SDK for Node.js

Gunakan anotasi untuk mencatat informasi pada segmen atau subsegmen yang ingin diindeks untuk pencarian.

Persyaratan Anotasi

- Tombol — Kunci untuk anotasi X-Ray dapat memiliki hingga 500 karakter alfanumerik. Anda tidak dapat menggunakan spasi atau simbol selain titik atau titik (.)
- Nilai — Nilai untuk anotasi X-Ray dapat memiliki hingga 1.000 karakter Unicode.
- Jumlah Anotasi — Anda dapat menggunakan hingga 50 anotasi per jejak.

Untuk mencatat anotasi

1. Dapatkan referensi ke segmen atau subsegmen saat ini.

```
var AWSXRay = require('aws-xray-sdk');  
...  
var document = AWSXRay.getSegment();
```

2. Panggil `addAnnotation` dengan kunci String, serta nilai Boolean, Nomor, atau String.

```
document.addAnnotation("mykey", "my value");
```

Contoh berikut menunjukkan cara memanggil `putAnnotation` dengan kunci String yang mencakup titik, dan nilai Boolean, Number, atau String.

```
document.putAnnotation("testkey.test", "my value");
```

SDK mencatat penjelasan sebagai pasangan nilai kunci dalam objek `annotations` di dokumen segmen. Memanggil `addAnnotation` dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Untuk menemukan penelusuran yang memiliki anotasi dengan nilai-nilai tertentu, gunakan kata kunci `annotation[key]` dalam [ekspresi filter](#).

Example `app.js` - anotasi

```
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) {
    ...
  });
});
```

Mencatat metadata dengan X-Ray SDK for Node.js

Gunakan metadata untuk mencatat informasi pada segmen atau subsegmen yang tidak perlu diindeks untuk pencarian. Nilai metadata dapat berupa string, angka, Boolean, atau objek yang dapat diserialisasikan ke dalam objek atau baris JSON.

Untuk mencatat metadata

1. Dapatkan referensi ke segmen atau subsegmen saat ini.

```
var AWSXRay = require('aws-xray-sdk');  
...  
var document = AWSXRay.getSegment();
```

2. Panggil `addMetadata` dengan kunci String, Boolean, Nomor, String, atau nilai objek, dan namespace string.

```
document.addMetadata("my key", "my value", "my namespace");
```

atau

Panggil `addMetadata` hanya dengan kunci dan nilai.

```
document.addMetadata("my key", "my value");
```

Jika Anda tidak menentukan namespace, SDK menggunakan default. Memanggil `addMetadata` dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Merekam pengguna IDs dengan X-Ray SDK untuk Node.js

Rekam segmen pengguna IDs berdasarkan permintaan untuk mengidentifikasi pengguna yang mengirim permintaan. Operasi ini tidak kompatibel dengan AWS Lambda fungsi karena segmen di lingkungan Lambda tidak dapat diubah. Panggilan `setUser` dapat diterapkan hanya untuk segmen, bukan subsegment.

Untuk merekam pengguna IDs

1. Dapatkan referensi ke segmen atau subsegmen saat ini.

```
var AWSXRay = require('aws-xray-sdk');  
...  
var document = AWSXRay.getSegment();
```

2. Panggil `setUser()` dengan ID String pengguna yang mengirim permintaan.

```
var user = 'john123';  
  
AWSXRay.getSegment().setUser(user);
```

Anda dapat memanggil `setUser` untuk mencatat ID pengguna segera setelah aplikasi mulai memproses permintaan. Jika Anda akan menggunakan segmen untuk mengatur ID pengguna, Anda dapat mengaitkan panggilan dalam satu baris.

Example `app.js` - ID pengguna

```
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) {  
    ...  
  })  
});
```

Untuk menemukan penelusuran pada ID pengguna, gunakan kata kunci `user` dalam [ekspresi filter](#).

Bekerja menggunakan Python

Ada dua cara untuk instrumen aplikasi Python Anda untuk mengirim jejak ke X-Ray:

- [AWS Distro untuk OpenTelemetry Python](#) — Distribusi AWS yang menyediakan serangkaian pustaka open source untuk mengirim metrik dan jejak yang berkorelasi ke beberapa solusi AWS pemantauan, termasuk Amazon,, dan OpenSearch Amazon Service CloudWatch AWS X-Ray, melalui Distro for Collector.AWS OpenTelemetry
- [AWS X-Ray SDK untuk Python](#) - Satu set perpustakaan untuk menghasilkan dan mengirim jejak ke X-Ray melalui daemon X-Ray.

Untuk informasi selengkapnya, lihat [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#).

AWS Distro untuk Python OpenTelemetry

Dengan AWS Distro for OpenTelemetry (ADOT) Python, Anda dapat menginstruksikan aplikasi Anda sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan termasuk Amazon,, CloudWatch dan Amazon Service. AWS X-Ray OpenSearch Menggunakan X-Ray dengan ADOT memerlukan dua komponen: OpenTelemetry SDK diaktifkan untuk digunakan dengan X-Ray, dan AWS Distro untuk OpenTelemetry Kolektor diaktifkan untuk digunakan dengan X-Ray. ADOT Python menyertakan dukungan instrumentasi otomatis, memungkinkan aplikasi Anda mengirim jejak tanpa perubahan kode.

Untuk memulai, lihat dokumentasi [AWS Distro untuk OpenTelemetry Python](#).

Untuk informasi selengkapnya tentang menggunakan AWS Distro untuk OpenTelemetry with AWS X-Ray dan lainnya Layanan AWS, lihat [AWS Distro untuk OpenTelemetry atau Distro untuk AWS Dokumentasi](#). OpenTelemetry

Untuk informasi selengkapnya tentang dukungan dan penggunaan bahasa, lihat [AWS Observabilitas di GitHub](#).

AWS X-Ray SDK untuk Python

X-Ray SDK for Python adalah pustaka untuk aplikasi web Python yang menyediakan kelas dan metode untuk menghasilkan dan mengirim data pelacakan ke daemon X-Ray. Data pelacakan

mencakup informasi tentang permintaan HTTP masuk yang disajikan oleh aplikasi, dan panggilan yang dilakukan aplikasi ke layanan hilir menggunakan AWS SDK, klien HTTP, atau konektor database SQL. Anda juga dapat membuat segmen secara manual dan menambahkan informasi debug dalam anotasi dan metadata.

Anda dapat mengunduh SDK dengan pip.

```
$ pip install aws-xray-sdk
```

Note

X-Ray SDK for Python adalah proyek sumber terbuka. Anda dapat mengikuti proyek dan mengirimkan masalah dan menarik permintaan di GitHub: github.com/aws/aws-xray-sdk-python

Jika Anda menggunakan Django atau Flask, mulai dengan [menambahkan middleware SDK ke aplikasi Anda](#) untuk melacak permintaan yang masuk. middleware menciptakan [segmen](#) untuk setiap permintaan yang dilacak, dan melengkapi segmen ketika respons dikirim. Ketika segmen terbuka, Anda dapat menggunakan metode klien SDK untuk menambahkan informasi ke segmen dan membuat subsegmen untuk penelusuran panggilan hilir. SDK juga secara otomatis mencatat pengecualian yang aplikasi Anda lempar ketika segmen terbuka. Untuk aplikasi lain, Anda dapat [buat segmen secara manual](#).

Untuk fungsi Lambda yang disebut oleh aplikasi atau layanan yang diinstrumentasi, Lambda membaca [tracing header](#) dan pelacakan sampel permintaan secara otomatis. Untuk fungsi lainnya, Anda dapat [mengonfigurasi Lambda](#) untuk sampel dan pelacakan permintaan masuk. Dalam kedua kasus, Lambda membuat segmen dan menyediakannya ke X-Ray SDK.

Note

Pada Lambda, X-Ray SDK adalah opsional. Jika Anda tidak menggunakannya dalam fungsi Anda, peta layanan Anda masih akan menyertakan simpul untuk layanan Lambda, dan satu untuk setiap fungsi Lambda. Dengan menambahkan SDK, Anda dapat melakukan instrumen kode fungsi Anda untuk menambahkan subsegmen ke segmen fungsi yang dicatat oleh Lambda. Lihat [AWS Lambda dan AWS X-Ray](#) untuk informasi selengkapnya.

Lihat [Pekerja](#) untuk contoh fungsi Python yang diinstrumentasikan di Lambda.

Selanjutnya, gunakan X-Ray SDK for Python untuk instrumen panggilan hilir dengan [patching pustaka aplikasi Anda](#). SDK mendukung pustaka berikut.

Pustaka yang didukung

- [botocore](#), [boto3](#) — AWS SDK for Python (Boto) Klien instrumen.
- [pynamodb](#) – Versi PynamoDB Instrumen dari klien Amazon DynamoDB.
- [aiobotocore](#), [aioboto3](#) – Instrumen [asyncio](#)-versi terintegrasi dari klien SDK for Python.
- [requests](#), [aiohttp](#) – Instrumen klien HTTP tingkat tinggi.
- [httplib](#), [http.client](#) – Instrumen klien HTTP tingkat rendah dan pustaka tingkat yang lebih tinggi yang menggunakannya.
- [sqlite3](#)— SQLite Klien instrumen.
- [mysql-connector-python](#) – Instrumen klien MySQL.
- [pg8000](#) – Instrumen antarmuka Pure-Python PostgreSQL.
- [psycopg2](#) – Adaptor basis data Instrumen PostgreSQL.
- [pymongo](#) – Instrumen klien MongoDB.
- [pymysql](#)— Instrumen klien berbasis PyMy SQL untuk MySQL dan MariaDB.

Setiap kali aplikasi Anda membuat panggilan ke AWS, database SQL, atau layanan HTTP lainnya, SDK mencatat informasi tentang panggilan dalam subsegmen. Layanan AWS dan sumber daya yang Anda akses dalam layanan muncul sebagai node hilir pada peta jejak untuk membantu Anda mengidentifikasi kesalahan dan masalah pembatasan pada koneksi individual.

Setelah Anda mulai menggunakan SDK, sesuaikan perilakunya dengan [mengonfigurasi perekam dan middleware](#). Anda dapat menambahkan plugin untuk mencatat data mengenai sumber daya komputasi yang berjalan di aplikasi Anda, menyesuaikan perilaku sampling dengan mendefinisikan aturan sampling, dan mengatur tingkat log untuk melihat lebih atau kurang informasi dari SDK dalam log aplikasi Anda.

Catat informasi tambahan tentang permintaan dan pekerjaan yang dilakukan aplikasi Anda dalam [anotasi dan metadata](#). Anotasi adalah pasangan kunci-nilai sederhana yang diindeks untuk digunakan dengan [ekspresi filter](#), sehingga Anda dapat mencari pelacakan yang berisi data tertentu. Entri metadata kurang bersifat membatasi dan dapat mencatat seluruh objek dan array — segala yang dapat disambungkan ke dalam JSON.

Anotasi dan Metadata

Anotasi dan metadata adalah teks abritari yang Anda tambahkan ke segmen dengan X-Ray SDK. Anotasi diindekskan untuk digunakan dengan ekspresi filter. Metadata tidak diindeks, tetapi dapat dilihat di segmen mentah dengan konsol X-Ray atau API. Siapa pun yang Anda berikan akses baca ke X-Ray dapat melihat data ini.

Bila Anda memiliki banyak klien diinstrumentasi dalam kode Anda, segmen permintaan tunggal dapat berisi sejumlah besar subsegmen, satu untuk setiap panggilan yang dilakukan dengan klien yang diinstrumentasi. Anda dapat mengatur dan grup subsegmen dengan membungkus panggilan klien di [subsegmen kustom](#). Anda dapat membuat subsegmen kustom untuk seluruh fungsi atau bagian kode apa pun. Anda kemudian dapat mencatat metadata dan anotasi pada subsegmen bukan menulis segala sesuatu pada segmen induk.

Untuk dokumentasi referensi untuk kelas SDK dan metode, lihat [AWS X-Ray SDK for Python untuk Referensi API](#).

Persyaratan

X-Ray SDK for Python mendukung versi bahasa dan pustaka berikut.

- Python – 2.7, 3.4, dan yang lebih baru
- Django – 1.10 dan yang lebih baru
- Flask – 0.10 dan yang lebih baru
- aiohttp – 2.3.0 dan versi terbaru
- AWS SDK for Python (Boto) – 1.4.0 dan yang lebih baru
- botocore – 1.5.0 dan yang lebih baru
- enum – 0.4.7 dan yang lebih baru, untuk Python versi 3.4.0 dan yang lebih lama
- jsonpickle – 1.0.0 dan yang lebih baru
- setuptools – 40.6.3 dan yang lebih baru
- wrapt – 1.11.0 dan yang lebih baru

Manajemen dependensi

X-Ray SDK for Python tersedia dari pip.

- Package - `aws-xray-sdk`

Tambahkan SDK sebagai dependensi di file `requirements.txt` Anda.

Example `requirements.txt`

```
aws-xray-sdk==2.4.2
boto3==1.4.4
botocore==1.5.55
Django==1.11.3
```

Jika Anda menggunakan Elastic Beanstalk untuk men-deploy aplikasi Anda, Elastic Beanstalk menginstal semua paket di `requirements.txt` secara otomatis.

Mengonfigurasi X-Ray SDK for Python

X-Ray SDK for Python mencakup kelas bernama `xray_recorder` yang menyediakan pencatat global. Anda dapat mengonfigurasi pencatat global untuk menyesuaikan middleware yang membuat segmen untuk panggilan HTTP masuk.

Bagian-bagian

- [Plugin layanan](#)
- [Aturan pengambilan sampel](#)
- [Pencatatan log](#)
- [Konfigurasi pencatat dalam kode](#)
- [Konfigurasi pencatat dengan Django](#)
- [Variabel-variabel lingkungan](#)

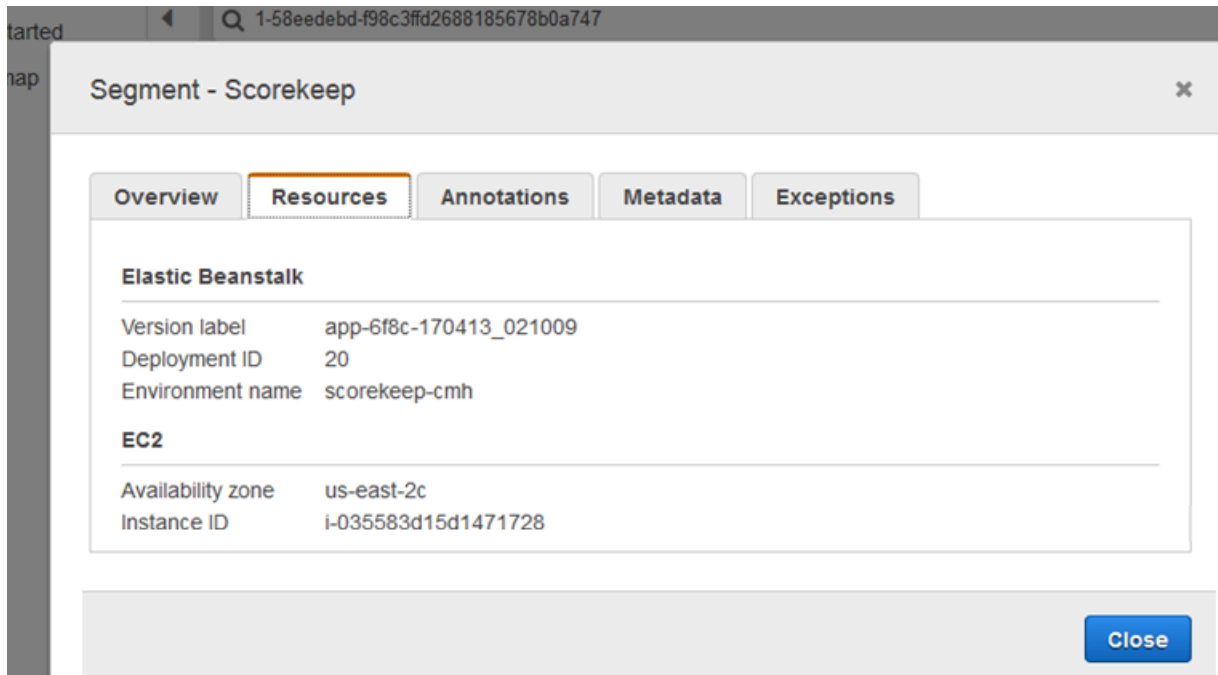
Plugin layanan

Gunakan `plugins` untuk mencatat informasi tentang layanan yang meng-hosting aplikasi Anda.

Plugin

- Amazon EC2 — `EC2Plugin` menambahkan ID instans, Availability Zone, dan Grup CloudWatch Log.

- Elastic Beanstalk – ElasticBeanstalkPlugin menambahkan nama lingkungan, label versi, dan ID deployment.
- Amazon ECS – ECSPugin menambahkan ID kontainer.



Untuk menggunakan plugin, hubungi configure di 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

Karena plugins diteruskan sebagai tuple, pastikan untuk menyertakan , saat menentukan plugin tunggal. Sebagai contoh, `plugins = ('EC2Plugin',)`.

Anda juga dapat menggunakan [variabel lingkungan](#), yang diutamakan daripada nilai yang ditetapkan dalam kode, untuk mengonfigurasi pencatat.

Konfigurasi plugin sebelum [pustaka patch](#) untuk mencatat panggilan hilir.

SDK juga menggunakan pengaturan plugin untuk mengatur bidang `origin` pada segmen. Ini menunjukkan jenis AWS sumber daya yang menjalankan aplikasi Anda. Saat Anda menggunakan beberapa plugin, SDK menggunakan urutan resolusi berikut untuk menentukan asal: ElasticBeanstalk > EKS > ECS > EC2

Aturan pengambilan sampel

SDK menggunakan aturan pengambilan sampel yang Anda tetapkan di konsol X-Ray untuk menentukan permintaan yang akan dicatat. Aturan default menelusuri permintaan pertama setiap detik, dan lima persen permintaan tambahan di semua layanan yang mengirim pelacakan ke X-Ray. [Buat aturan tambahan di konsol X-Ray](#) untuk menyesuaikan jumlah data yang dicatat untuk setiap aplikasi Anda.

SDK menerapkan aturan kustom sesuai urutan penetapannya. Jika permintaan cocok dengan beberapa aturan kustom, SDK hanya menerapkan aturan pertama.

Note

Jika SDK tidak dapat mencapai X-Ray untuk mendapatkan aturan pengambilan sampel, SDK akan beralih ke aturan lokal default dari permintaan pertama setiap detik, dan lima persen permintaan tambahan per host. Hal ini dapat terjadi jika host tidak memiliki izin untuk memanggil sampling APIs, atau tidak dapat terhubung ke daemon X-Ray, yang bertindak sebagai proxy TCP untuk panggilan API yang dilakukan oleh SDK.

Anda juga dapat mengonfigurasi SDK untuk memuat aturan sampling dari dokumen JSON. SDK dapat menggunakan aturan lokal sebagai cadangan jika terjadi kasus tidak dapat mengambil sampel X-Ray, atau menggunakan aturan lokal secara eksklusif.

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
  }
}
```

Contoh ini menentukan satu aturan kustom dan aturan default. Aturan kustom menerapkan tingkat pengambilan sampel lima persen tanpa jumlah minimum permintaan untuk melacak jalur di `/api/move/`. Aturan default menelusuri permintaan pertama setiap detik dan 10 persen dari permintaan tambahan.

Kerugian dari menentukan aturan secara lokal adalah bahwa target tetap diterapkan oleh setiap instans pencatat secara independen, alih-alih dikelola oleh layanan X-Ray. Ketika Anda men-deploy lebih banyak host, laju tetap akan dikalikan, sehingga sulit untuk mengontrol jumlah data yang dicatat.

Pada AWS Lambda, Anda tidak dapat mengubah laju pengambilan sampel. Jika fungsi Anda dipanggil oleh layanan yang diinstrumentasikan, panggilan yang menghasilkan permintaan yang sampelnya diambil oleh layanan yang akan dicatat oleh Lambda. Jika pelacakan aktif diaktifkan dan tidak ada header pelacakan, Lambda membuat keputusan pengambilan sampel.

Untuk mengonfigurasi aturan pengambilan sampel cadangan, panggil `xray_recorder.configure`, seperti yang ditunjukkan pada contoh berikut, di mana *rules* adalah kamus aturan atau jalur absolut ke file JSON yang berisi aturan pengambilan sampel.

```
xray_recorder.configure(sampling_rules=rules)
```

Untuk menggunakan hanya aturan lokal, konfigurasi pencatat dengan `LocalSampler`.

```
from aws_xray_sdk.core.sampling.local.sampler import LocalSampler
xray_recorder.configure(sampler=LocalSampler())
```

Anda juga dapat mengonfigurasi pencatat global untuk menonaktifkan pengambilan sampel dan instrumen semua permintaan masuk.

Example main.py - Nonaktifkan sampling

```
xray_recorder.configure(sampling=False)
```

Pencatatan log

SDK menggunakan modul logging Python bawaan dengan tingkat pencatatan WARNING default. Dapatkan referensi ke pencatat untuk kelas `aws_xray_sdk` dan panggil `setLevel` untuk mengonfigurasi tingkat log yang berbeda untuk pustaka dan sisa aplikasi Anda.

Example app.py - Pencatatan

```
logging.basicConfig(level='WARNING')  
logging.getLogger('aws_xray_sdk').setLevel(logging.ERROR)
```

Gunakan log debug untuk mengidentifikasi masalah, seperti subsegmen yang tidak tertutup, saat Anda [menghasilkan subsegmen secara manual](#).

Konfigurasi pencatat dalam kode

Pengaturan tambahan tersedia dari metode `configure` pada `xray_recorder`.

- `context_missing` – Mengatur ke `LOG_ERROR` untuk menghindari mengembalikan pengecualian ketika kode instrumentasi Anda mencoba mencatat data ketika tidak ada segmen yang terbuka.
- `daemon_address` – Mengatur host dan port listener daemon X-Ray.
- `service` – Mengatur nama layanan yang digunakan SDK untuk segmen.
- `plugins`— Rekam informasi tentang AWS sumber daya aplikasi Anda.
- `sampling` – Atur ke `False` untuk menonaktifkan pengambilan sampel.
- `sampling_rules` – Mengatur jalur dari file JSON yang berisi [Aturan pengambilan sampel](#).

Example main.py - Nonaktifkan pengecualian konteks yang hilang

```
from aws_xray_sdk.core import xray_recorder  
  
xray_recorder.configure(context_missing='LOG_ERROR')
```

Konfigurasi pencatat dengan Django

Jika Anda menggunakan kerangka kerja Django, Anda dapat menggunakan file `settings.py` Django untuk mengonfigurasi opsi pada pencatat global.

- `AUTO_INSTRUMENT` (hanya Django) – Mencatat subsegmen untuk basis data bawaan dan operasi penyajian templat.
- `AWS_XRAY_CONTEXT_MISSING` – Mengatur ke `LOG_ERROR` untuk menghindari mengembalikan pengecualian ketika kode instrumentasi Anda mencoba untuk mencatat data ketika tidak ada segmen yang terbuka.
- `AWS_XRAY_DAEMON_ADDRESS` – Mengatur host dan port listener daemon X-Ray.
- `AWS_XRAY_TRACING_NAME` – Mengatur nama layanan yang digunakan SDK untuk segmen.
- `PLUGINS`— Rekam informasi tentang AWS sumber daya aplikasi Anda.
- `SAMPLING` – Atur ke `False` untuk menonaktifkan pengambilan sampel.
- `SAMPLING_RULES` – Mengatur jalur dari file JSON yang berisi [Aturan pengambilan sampel](#).

Untuk mengaktifkan konfigurasi pencatat di `settings.py`, tambahkan middleware Django ke daftar aplikasi terinstal.

Example `settings.py` – Aplikasi terinstal

```
INSTALLED_APPS = [  
    ...  
    'django.contrib.sessions',  
    'aws_xray_sdk.ext.django',  
]
```

Mengonfigurasi pengaturan yang tersedia dalam kamus bernama `XRAY_RECORDER`.

Example `settings.py` – Aplikasi terinstal

```
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,
```



```
}
```

Variabel-variabel lingkungan

Anda dapat menggunakan variabel lingkungan untuk mengonfigurasi X-Ray SDK for Python. SDK mendukung variabel berikut:

- `AWS_XRAY_TRACING_NAME` – Mengatur nama layanan yang digunakan SDK untuk segmen. Menimpa nama layanan yang Anda tetapkan secara terprogram.
- `AWS_XRAY_SDK_ENABLED` – Ketika diatur ke `false`, menonaktifkan SDK. Secara default, SDK diaktifkan kecuali variabel lingkungan diatur ke salah.
 - Ketika dinonaktifkan, pencatat global secara otomatis menghasilkan segmen dan subsegmen dummy yang tidak dikirim ke daemon, dan patch otomatis dinonaktifkan. Middlewares ditulis sebagai pembungkus atas pencatat global. Semua segmen dan subsegmen generasi melalui middleware juga menjadi segmen dummy dan subsegment dummy.
 - Menetapkan nilai `AWS_XRAY_SDK_ENABLED` melalui variabel lingkungan atau melalui interaksi langsung dengan objek `global_sdk_config` dari pustaka `aws_xray_sdk`. Pengaturan ke variabel lingkungan menimpa interaksi ini.
- `AWS_XRAY_DAEMON_ADDRESS` – Mengatur host dan port listener daemon X-Ray. Secara default, SDK menggunakan `127.0.0.1:2000` untuk data pelacakan (UDP) dan pengambilan sampel (TCP). Gunakan variabel ini jika Anda telah mengonfigurasi daemon untuk [mendengarkan di port berbeda](#) atau jika berjalan pada host yang berbeda.

format

- Port yang sama – `address:port`
- Port yang berbeda – `tcp:address:port udp:address:port`
- `AWS_XRAY_CONTEXT_MISSING`— Setel `RUNTIME_ERROR` untuk melempar pengecualian saat kode instrumentasi Anda mencoba merekam data saat tidak ada segmen yang terbuka.

Nilai Valid

- `RUNTIME_ERROR`— Lempar pengecualian runtime.
- `LOG_ERROR`— Log kesalahan dan lanjutkan (default).
- `IGNORE_ERROR`— Abaikan kesalahan dan lanjutkan.

Kesalahan yang berkaitan dengan segmen atau subsegmen yang hilang dapat terjadi ketika Anda mencoba untuk menggunakan klien yang diinstrumentasi dalam kode perusahaan rintisan yang berjalan ketika tidak ada permintaan terbuka, atau dalam kode yang memunculkan thread baru.

variabel lingkungan menimpa nilai yang ditetapkan dalam kode.

Menelusuri permintaan yang masuk dengan perangkat tengah SDK for Python X-Ray

Ketika Anda menambahkan perangkat tengah ke aplikasi Anda dan mengonfigurasi nama segmen, SDK for Python X-Ray membuat segmen untuk setiap permintaan sampel. Segmen ini mencakup waktu, metode, dan disposisi permintaan HTTP. Instrumentasi tambahan membuat subsegmen pada segmen ini.

SDK for Python X-Ray mendukung perangkat tengah berikut ini untuk instrumen permintaan HTTP yang masuk:

- Django
- Flask
- Botol

Note

Untuk AWS Lambda fungsi, Lambda membuat segmen untuk setiap permintaan sampel. Lihat [AWS Lambda dan AWS X-Ray](#) untuk informasi selengkapnya.

Lihat [Pekerja](#) untuk contoh fungsi Python yang diinstrumentasikan di Lambda.

Untuk skrip atau aplikasi Python pada kerangka kerja lain, Anda dapat [membuat segmen secara manual](#).

Setiap segmen memiliki nama yang mengidentifikasi aplikasi Anda dalam peta layanan. Segmen dapat diberi nama secara statis, atau Anda dapat mengonfigurasi SDK untuk nama itu secara dinamis berdasarkan header host dalam permintaan masuk. Penamaan dinamis memungkinkan Anda mengelompokkan pelacakan berdasarkan nama domain dalam permintaan, dan menerapkan

nama default jika nama tersebut tidak cocok dengan pola yang diharapkan (misalnya, jika header `host` ditiru).

Permintaan yang Diteruskan

Jika penyeimbang beban atau perantara lainnya meneruskan permintaan ke aplikasi Anda, X-Ray akan mengambil IP klien dari header `X-Forwarded-For` dalam permintaan bukan dari sumber IP dalam paket IP. IP klien yang dicatat untuk permintaan yang diteruskan dapat ditiru, sehingga tidak dapat dipercaya.

Ketika permintaan diteruskan, SDK menetapkan bidang tambahan di segmen untuk menunjukkan ini. Jika segmen berisi bidang `x_forwarded_for` atur `true`, IP klien diambil dari header `X-Forwarded-For` dalam permintaan HTTP.

Perangkat tengah membuat segmen untuk setiap permintaan masuk dengan blok `http` yang berisi informasi berikut:

- Metode HTTP – GET, POST, PUT, DELETE, dll
- Alamat klien – Alamat IP klien yang mengirim permintaan.
- Kode respons – Kode respons HTTP untuk permintaan yang selesai.
- Timing – Waktu mulai (saat permintaan diterima) dan waktu akhir (saat respons dikirim).
- Agen pengguna — `user-agent` dari permintaan.
- Panjang konten — `content-length` dari respons.

Bagian-bagian

- [Menambahkan perangkat tengah ke aplikasi Anda \(Django\)](#)
- [Menambahkan perangkat tengah ke aplikasi Anda \(flask\)](#)
- [Menambahkan perangkat tengah ke aplikasi Anda \(Bottle\)](#)
- [Menginstrumentasikan kode Python secara manual](#)
- [Mengonfigurasi strategi penamaan segmen](#)

Menambahkan perangkat tengah ke aplikasi Anda (Django)

Tambahkan perangkat tengah ke daftar MIDDLEWARE di file `settings.py` Anda. Perangkat tengah X-Ray harus menjadi baris pertama dalam file `settings.py` untuk memastikan bahwa permintaan yang gagal di perangkat tengah lain dicatat.

Example `settings.py` - perangkat tengah SDK for Python X-Ray

```
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'  
]
```

Tambahkan aplikasi Django SDK X-Ray ke daftar `INSTALLED_APPS` di file `settings.py`. Tindakan ini akan mengizinkan pencatat X-Ray dikonfigurasi selama memulai aplikasi Anda.

Example `settings.py` - aplikasi Django SDK for Python X-Ray

```
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',  
]
```

Mengonfigurasi nama segmen di [file `settings.py`](#).

Example `settings.py` - Nama segmen

```
XRAY_RECORDER = {  
    'AWS_XRAY_TRACING_NAME': 'My application',  
    'PLUGINS': ('EC2Plugin',),  
}
```

Ini memberitahu pencatat X-Ray untuk melacak permintaan yang dilayani oleh aplikasi Django Anda dengan tingkat pengambilan sampel default. Anda dapat [mengonfigurasi pencatat file pengaturan Django Anda](#) untuk menerapkan aturan pengambilan sampel kustom atau mengubah pengaturan lainnya.

Note

Karena plugins diteruskan sebagai tuple, pastikan untuk menyertakan `,` saat menentukan plugin tunggal. Sebagai contoh, `plugins = ('EC2Plugin',)`.

Menambahkan perangkat tengah ke aplikasi Anda (flask)

Untuk instrumen aplikasi Flask Anda, pertama mengonfigurasi nama segmen pada `xray_recorder`. Kemudian, gunakan fungsi `XRayMiddleware` untuk mem-patch aplikasi Flask Anda dalam kode.

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)
```

Ini memberitahu pencatat X-Ray untuk melacak permintaan yang disediakan oleh aplikasi Flask Anda dengan tingkat pengambilan sampel default. Anda dapat [mengonfigurasi pencatat dalam kode](#) untuk menerapkan aturan pengambilan sampel kustom atau mengubah pengaturan lainnya.

Menambahkan perangkat tengah ke aplikasi Anda (Bottle)

Untuk instrumen aplikasi Bottle Anda, pertama-tama konfigurasikan nama segmen pada `xray_recorder`. Kemudian, gunakan fungsi `XRayMiddleware` untuk melakukan patch aplikasi Bottle Anda dalam kode.

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))
```

Ini memberitahu pencatat X-Ray untuk melacak permintaan yang disediakan oleh aplikasi Bottle Anda dengan tingkat pengambilan sampel default. Anda dapat [mengonfigurasi pencatat dalam kode](#) untuk menerapkan aturan pengambilan sampel kustom atau mengubah pengaturan lainnya.

Menginstrumentasikan kode Python secara manual

Jika Anda tidak menggunakan Django atau Flask, Anda dapat membuat segmen secara manual. Anda dapat membuat segmen untuk setiap permintaan yang masuk, atau membuat segmen di sekitar klien HTTP atau AWS SDK yang ditambah untuk menyediakan konteks bagi perekam untuk menambahkan subsegmen.

Example main.py - instrumentasi manual

```
from aws_xray_sdk.core import xray_recorder

# Start a segment
segment = xray_recorder.begin_segment('segment_name')
# Start a subsegment
subsegment = xray_recorder.begin_subsegment('subsegment_name')

# Add metadata and annotations
segment.put_metadata('key', dict, 'namespace')
subsegment.put_annotation('key', 'value')

# Close the subsegment and segment
xray_recorder.end_subsegment()
xray_recorder.end_segment()
```

Mengonfigurasi strategi penamaan segmen

AWS X-Ray menggunakan nama layanan untuk mengidentifikasi aplikasi Anda dan membedakannya dari aplikasi lain, database, eksternal APIs, dan AWS sumber daya yang digunakan aplikasi Anda. Saat SDK X-Ray membuat segmen untuk permintaan masuk, SDK akan mencatat nama layanan aplikasi Anda di [kolom nama](#).

SDK X-Ray dapat memberi nama segmen setelah nama host di header permintaan HTTP. Namun, header ini dapat ditiru, yang dapat mengakibatkan simpul tak terduga di peta layanan Anda. Untuk

mencegah SDK dari penamaan segmen salah karena permintaan dengan header host palsu, Anda harus menentukan nama default untuk permintaan masuk.

Jika aplikasi Anda menyuguhkan permintaan untuk beberapa domain, Anda dapat mengonfigurasi SDK untuk menggunakan strategi penamaan dinamis untuk mencerminkan ini dalam nama segmen. Strategi penamaan dinamis mengizinkan SDK menggunakan nama host untuk permintaan yang sesuai dengan pola yang diharapkan, dan menerapkan nama default untuk permintaan yang tidak sesuai.

Misalnya, Anda boleh memiliki satu aplikasi yang melayani permintaan untuk tiga subdomain—`www.example.com`, `api.example.com`, dan `static.example.com`. Anda dapat menggunakan strategi penamaan dinamis dengan pola `*.example.com` untuk mengidentifikasi segmen untuk setiap subdomain dengan nama yang berbeda, mengakibatkan tiga simpul layanan pada peta layanan. Jika aplikasi Anda menerima permintaan dengan nama host yang tidak cocok dengan pola, Anda akan melihat simpul keempat pada peta layanan dengan nama fallback yang Anda tentukan.

Untuk menggunakan nama yang sama untuk semua segmen permintaan, tentukan nama aplikasi Anda ketika mengonfigurasi pencatat, seperti yang ditunjukkan di [bagian sebelumnya](#).

Strategi penamaan dinamis menentukan pola yang harus sesuai dengan nama host, dan nama default untuk digunakan jika nama host dalam permintaan HTTP tidak cocok dengan pola. Untuk menamai segmen secara dinamis di Django, tambahkan pengaturan `DYNAMIC_NAMING` ke file [settings.py](#).

Example settings.py - Penamaan dinamis

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_TRACING_NAME': 'My application',
    'DYNAMIC_NAMING': '*.example.com',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin')
}
```

Anda dapat menggunakan `*` dalam pola untuk mencocokkan string apa pun, atau `?` untuk mencocokkan setiap karakter tunggal. Untuk Flask, [konfigurasi pencatat dalam simpul](#).

Example main.py - Nama segmen

```
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='My application')
```

```
xray_recorder.configure(dynamic_naming='*.example.com')
```

Note

Anda dapat mengganti nama layanan default yang Anda tentukan dalam kode dengan [variabel lingkungan](#) `AWS_XRAY_TRACING_NAME`.

Pustaka patching ke instrumen panggilan hilir

Untuk instrumen panggilan hilir, gunakan X-Ray SDK for Python untuk patch pustaka yang menggunakan aplikasi Anda. X-Ray SDK for Python dapat patch pustaka berikut.

Pustaka yang didukung

- [botocore](#), [boto3](#) — AWS SDK for Python (Boto) Klien instrumen.
- [pynamodb](#) – Versi PynamoDB Instrumen dari klien Amazon DynamoDB.
- [aiobotocore](#), [aioboto3](#) – Instrumen [asyncio](#)-versi terintegrasi dari klien SDK for Python.
- [requests](#), [aiohttp](#) – Instrumen klien HTTP tingkat tinggi.
- [httplib](#), [http.client](#) – Instrumen klien HTTP tingkat rendah dan pustaka tingkat yang lebih tinggi yang menggunakannya.
- [sqlite3](#)— SQLite Klien instrumen.
- [mysql-connector-python](#) – Instrumen klien MySQL.
- [pg8000](#) – Instrumen antarmuka Pure-Python PostgreSQL.
- [psycopg2](#) – Adaptor basis data Instrumen PostgreSQL.
- [pymongo](#) – Instrumen klien MongoDB.
- [pymysql](#)— Instrumen klien berbasis PyMy SQL untuk MySQL dan MariaDB.

Ketika Anda menggunakan pustaka yang dipatch, X-Ray SDK for Python membuat subsegmen untuk panggilan dan catatan informasi dari permintaan dan respons. Segmen harus tersedia untuk SDK untuk membuat subsegmen, baik dari SDK middleware atau dari AWS Lambda.

 Note

Jika Anda menggunakan SQLAlchemy ORM, Anda dapat menginstrumentasikan kueri SQL Anda dengan mengimpor versi sesi dan kelas kueri SDK. SQLAlchemy Lihat [Gunakan SQLAlchemy ORM](#) untuk instruksi.

Untuk patch semua pustaka yang tersedia, gunakan fungsi `patch_all` di `aws_xray_sdk.core`. Beberapa pustaka, seperti `httplib` dan `urllib`, mungkin perlu mengaktifkan patch ganda dengan memanggil `patch_all(double_patch=True)`.

Example main.py - Patch semua pustaka yang didukung

```
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()
```

Untuk patch sebuah pustaka tunggal, hubungi patch dengan tupel dari nama pustaka. Untuk mencapai hal ini, Anda perlu menyediakan daftar elemen tunggal.

Example main.py - Patch pustaka tertentu

```
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

Dalam beberapa kasus, kunci yang Anda gunakan untuk patch pustaka tidak cocok dengan nama pustaka. Beberapa kunci berfungsi sebagai alias untuk satu atau lebih pustaka.

Alias Pustaka

- `httplib`— [httplib](#) dan [http.client](#)
- `mysql` – [mysql-connector-python](#)

Menelusuri konteks untuk pekerjaan asynchronous

Untuk pustaka terintegrasi `asyncio`, atau untuk [membuat subsegment bagi fungsi asynchronous](#), Anda juga harus mengonfigurasi X-Ray SDK for Python dengan konteks `async`. Mengimpor kelas `AsyncContext` dan lulus instans nya untuk pencatat X-Ray.

Note

Pustaka dukungan kerangka kerja web, seperti `AIOHTTP`, tidak ditangani melalui modul `aws_xray_sdk.core.patcher`. Mereka tidak akan muncul dalam katalog `patcher` dari pustaka yang didukung.

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)
```

Menelusuri panggilan AWS SDK dengan X-Ray SDK untuk Python

[Saat aplikasi Anda melakukan panggilan Layanan AWS untuk menyimpan data, menulis ke antrian, atau mengirim notifikasi, X-Ray SDK untuk Python melacak panggilan hilir di subsegmen.](#) Ditelusuri Layanan AWS dan sumber daya yang Anda akses dalam layanan tersebut (misalnya, bucket Amazon S3 atau antrian Amazon SQS), muncul sebagai node hilir pada peta jejak di konsol X-Ray.

X-Ray SDK untuk Python secara otomatis menginstrumentasikan AWS semua klien SDK saat [Anda](#) menambal perpustakaan. `botocore` Anda tidak dapat instrumen klien individu.

Untuk semua layanan, Anda dapat melihat nama panggilan API di konsol X-Ray. Untuk subset layanan, X-Ray SDK menambahkan informasi ke segmen untuk memberikan lebih banyak perincian di peta layanan.

Sebagai contoh, ketika Anda melakukan panggilan dengan klien DynamoDB berinstrumen, SDK menambahkan nama tabel ke segmen untuk panggilan yang menargetkan tabel. Di konsol tersebut, setiap tabel muncul sebagai simpul terpisah di peta layanan, dengan simpul DynamoDB generik untuk panggilan yang tidak menargetkan tabel.

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Ketika Anda mengakses sumber daya bernama, panggilan ke layanan berikut membuat simpul tambahan di peta layanan. Panggilan yang tidak menargetkan sumber daya tertentu membuat simpul generik untuk layanan tersebut.

- Amazon DynamoDB – Nama tabel
- Amazon Simple Storage Service – Bucket dan nama kunci
- Amazon Simple Queue Service – Nama antrean

Menelusuri panggilan ke layanan web hilir HTTP menggunakan X-Ray SDK for Python

Ketika aplikasi Anda melakukan panggilan ke microservices atau HTTP publik APIs, Anda dapat menggunakan X-Ray SDK untuk Python untuk instrumen panggilan tersebut dan menambahkan API ke grafik layanan sebagai layanan hilir.

Untuk klien HTTP instrumen, [patch pustaka](#) yang Anda gunakan untuk melakukan panggilan keluar. Jika Anda menggunakan `requests` atau Klien HTTP bawaan Python, itu semua yang perlu Anda lakukan. Untuk `aiohttp`, juga mengonfigurasi pencatat dengan [konteks asinkron](#).

Jika Anda menggunakan Klien API `aiohttp` 3, Anda juga perlu mengonfigurasi `ClientSession` dengan sebuah instans dari konfigurasi penelusuran yang disediakan oleh SDK.

Example [aiohttp3 klien API](#)

```
from aws_xray_sdk.ext.aiohttp.client import aws_xray_trace_config

async def foo():
    trace_config = aws_xray_trace_config()
    async with ClientSession(loop=loop, trace_configs=[trace_config]) as session:
        async with session.get(url) as resp:
            await resp.read()
```

Ketika Anda instrumen panggilan ke API web hilir, X-Ray SDK for Python mencatat subsegmen dengan informasi tentang permintaan HTTP dan respon. X-Ray menggunakan subsegmen untuk menghasilkan segmen disimpulkan untuk API jarak jauh.

Example Subsegmen untuk panggilan HTTP downstream

```
{
```

```

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

```

Example Segmen yang disimpulkan untuk panggilan HTTP downstream

```

{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}

```

Menghasilkan subsegment kustom dengan X-Ray SDK for Python

Subsegmen memperpanjang [segmen](#) penelusuran dengan detail tentang pekerjaan yang dilakukan untuk melayani permintaan. Setiap kali Anda melakukan panggilan dengan klien berinstrumen, X-Ray

tersebut mencatat informasi yang dihasilkan dalam subsegment. Anda dapat membuat subsegment tambahan untuk mengelompokkan subsegment lain, untuk mengukur performa bagian kode, atau untuk mencatat anotasi dan metadata.

Untuk mengelola subsegment, gunakan metode `begin_subsegment` dan `end_subsegment`.

Example main.py - Subsegmen kustom

```
from aws_xray_sdk.core import xray_recorder

subsegment = xray_recorder.begin_subsegment('annotations')
subsegment.put_annotation('id', 12345)
xray_recorder.end_subsegment()
```

Untuk membuat subsegmen untuk fungsi sinkron, gunakan dekorator `@xray_recorder.capture`. Anda dapat melewati nama untuk subsegment ke fungsi `capture` atau meninggalkannya agar menggunakan nama fungsi.

Example main.py - Fungsi subsegmen

```
from aws_xray_sdk.core import xray_recorder

@xray_recorder.capture('## create_user')
def create_user():
    ...
```

Untuk fungsi asynchronous, gunakan dekorator `@xray_recorder.capture_async`, dan lewati konteks `async` untuk mencatat.

Example main.py - fungsi asynchronous subsegmen

```
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()
```

Ketika Anda membuat subsegmen dalam segmen atau subsegmen lain, X-Ray SDK for Python membuat ID untuknya dan mencatat waktu mulai dan waktu akhir.

Example Subsegmen dengan metadata

```
"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"
    }
  },
},
```

Tambahkan anotasi dan metadata ke segmen dengan X-Ray SDK for Python

Anda dapat mencatat informasi tambahan tentang permintaan, lingkungan, atau aplikasi Anda dalam anotasi dan metadata. Anda dapat menambahkan anotasi dan metadata ke segmen yang dibuat oleh SDK X-Ray, atau subsegmen kustom yang Anda buat.

Anotasi adalah pasangan kunci-nilai dengan string, nomor, atau nilai-nilai Boolean. Anotasi diindekskan untuk digunakan dengan [Ekspresi filter](#). Gunakan anotasi untuk mencatat data yang ingin Anda gunakan untuk mengelompokkan pelacakan di konsol tersebut, atau saat memanggil API [GetTraceSummaries](#).

Metadata adalah pasangan kunci-nilai yang dapat memiliki nilai dari setiap tipe, termasuk objek dan daftar, tetapi tidak diindekskan untuk digunakan dengan ekspresi filter. Gunakan metadata untuk mencatat data tambahan yang ingin disimpan dalam pelacakan tetapi tidak perlu digunakan dengan pencarian.

Selain anotasi dan metadata, Anda juga dapat [mencatat string ID pengguna](#) pada segmen. Pengguna IDs direkam dalam bidang terpisah pada segmen dan diindeks untuk digunakan dengan pencarian.

Bagian-bagian

- [Anotasi pencatatan dengan X-Ray SDK for Python](#)

- [Metadata pencatatan dengan X-Ray SDK for Python](#)
- [Merekam pengguna IDs dengan X-Ray SDK untuk Python](#)

Anotasi pencatatan dengan X-Ray SDK for Python

Gunakan anotasi untuk mencatat informasi pada segmen atau subsegmen yang ingin diindeks untuk pencarian.

Persyaratan Anotasi

- Tombol — Kunci untuk anotasi X-Ray dapat memiliki hingga 500 karakter alfanumerik. Anda tidak dapat menggunakan spasi atau simbol selain titik atau titik (.).
- Nilai — Nilai untuk anotasi X-Ray dapat memiliki hingga 1.000 karakter Unicode.
- Jumlah Anotasi — Anda dapat menggunakan hingga 50 anotasi per jejak.

Untuk mencatat anotasi

1. Dapatkan referensi ke segmen atau subsegmen saat ini dari `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

atau

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. Panggil `put_annotation` dengan kunci String, serta nilai Boolean, Nomor, atau String.

```
document.put_annotation("mykey", "my value");
```

Contoh berikut menunjukkan cara memanggil `putAnnotation` dengan kunci String yang mencakup titik, dan nilai Boolean, Number, atau String.

```
document.putAnnotation("testkey.test", "my value");
```


Atau, Anda dapat menggunakan metode `put_annotation` pada `xray_recorder`. Metode ini mencatat penjelasan pada subsegmen saat ini atau, jika tidak ada subsegmen terbuka, pada segmen.

```
xray_recorder.put_annotation("mykey", "my value");
```

SDK mencatat penjelasan sebagai pasangan kunci-nilai dalam objek `annotations` dokumen segmen. Memanggil `put_annotation` dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Untuk menemukan penelusuran yang memiliki anotasi dengan nilai-nilai tertentu, gunakan kata kunci `annotation[key]` dalam [ekspresi filter](#).

Metadata pencatatan dengan X-Ray SDK for Python

Gunakan metadata untuk mencatat informasi pada segmen atau subsegmen yang tidak perlu diindeks untuk pencarian. Nilai metadata bisa string, angka, Boolean, atau objek yang dapat diserialkan ke dalam objek JSON atau array.

Untuk mencatat metadata

1. Dapatkan referensi ke segmen atau subsegmen saat ini dari `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

atau

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. Panggil `put_metadata` dengan kunci String, Boolean, Nomor, String, atau nilai Objek; dan namespace String.

```
document.put_metadata("my key", "my value", "my namespace");
```

atau

Panggil `put_metadata` dengan kunci dan nilai.

```
document.put_metadata("my key", "my value");
```

Atau, Anda dapat menggunakan metode `put_metadata` pada `xray_recorder`. Metode ini mencatat metadata pada subsegmen saat ini atau, jika tidak ada subsegmen terbuka, pada segmen.

```
xray_recorder.put_metadata("my key", "my value");
```

Jika Anda tidak menentukan namespace, SDK menggunakan default. Memanggil `put_metadata` dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Merekam pengguna IDs dengan X-Ray SDK untuk Python

Rekam segmen pengguna IDs berdasarkan permintaan untuk mengidentifikasi pengguna yang mengirim permintaan.

Untuk merekam pengguna IDs

1. Dapatkan referensi ke segmen saat ini dari `xray_recorder`.

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

2. Panggil `setUser` dengan ID String dari pengguna yang mengirim permintaan.

```
document.set_user("U12345");
```

Anda dapat menelepon `set_user` di pengendali Anda untuk mencatat ID pengguna segera setelah aplikasi Anda mulai memproses permintaan.

Untuk menemukan penelusuran pada ID pengguna, gunakan kata kunci `user` di [ekspresi filter](#).

Instrumenting kerangka kerja web dideploy ke lingkungan nserver

AWS X-Ray SDK untuk Python mendukung kerangka kerja web instrumentasi yang digunakan dalam aplikasi tanpa server. Nserver adalah arsitektur asli cloud yang memungkinkan Anda mengalihkan lebih banyak tanggung jawab operasional Anda ke AWS, meningkatkan ketangkasan dan inovasi Anda.

Arsitektur nserver adalah model aplikasi perangkat lunak yang memungkinkan Anda untuk membangun dan menjalankan aplikasi dan layanan tanpa memikirkan server. Menghilangkan tugas-tugas manajemen infrastruktur seperti server atau penyediaan klaster, patching, pemeliharaan sistem operasi, dan penyediaan kapasitas. Anda dapat membangun solusi tanpa server untuk hampir semua tipe aplikasi atau layanan backend, dan semua yang diperlukan untuk menjalankan dan menskalakan aplikasi Anda dengan ketersediaan tinggi ditangani untuk Anda.

Tutorial ini menunjukkan kepada Anda bagaimana untuk secara otomatis instrumen AWS X-Ray pada kerangka web, seperti Flask atau Django, yang dikerahkan ke lingkungan tanpa server. Instrumentasi X-Ray aplikasi memungkinkan Anda untuk melihat semua panggilan hilir yang dibuat, mulai dari Amazon API Gateway melalui AWS Lambda fungsi Anda, dan panggilan keluar yang dilakukan aplikasi Anda.

X-Ray SDK for Python mendukung kerangka kerja aplikasi Python berikut:

- Flask versi 0.8, atau lebih baru
- Django versi 1.0, atau yang lebih baru

Tutorial ini mengembangkan contoh aplikasi nserver yang dideploy ke Lambda dan dipanggil oleh API Gateway. Tutorial ini menggunakan Zappa untuk secara otomatis mendeploy aplikasi untuk Lambda dan mengonfigurasi titik akhir API Gateway.

Prasyarat

- [Zappa](#)
- [Python](#) – Versi 2.7 atau 3.6.
- [AWS CLI](#)— Verifikasi bahwa Anda AWS CLI dikonfigurasi dengan akun dan Wilayah AWS di mana Anda akan menyebarkan aplikasi Anda.
- [Pip](#)
- [Virtualenv](#)

Langkah 1: Buat lingkungan

Pada langkah ini, Anda membuat lingkungan virtual menggunakan `virtualenv` untuk meng-host aplikasi.

1. Menggunakan AWS CLI, buat direktori untuk aplikasi. Kemudian ubah ke direktori baru.

```
mkdir serverless_application  
cd serverless_application
```

2. Berikutnya, buat lingkungan virtual dalam direktori baru Anda. Gunakan perintah berikut untuk mengaktifkannya.

```
# Create our virtual environment  
virtualenv serverless_env  
  
# Activate it  
source serverless_env/bin/activate
```

3. Instal X-Ray, Flask, Zappa, dan pustaka Permintaan ke lingkungan Anda.

```
# Install X-Ray, Flask, Zappa, and Requests into your environment  
pip install aws-xray-sdk flask zappa requests
```

4. Tambahkan kode aplikasi ke `serverless_application` direktori. Contohnya, kita dapat membangun dari contoh Flasks [Halo Dunia](#).

Di direktori `serverless_application`, buat file baru bernama `my_app.py`. Kemudian gunakan editor teks untuk menambahkan perintah berikut. Aplikasi ini instrumen pustaka Permintaan, patch middleware aplikasi Flask ini, dan membuka titik akhir `'/'`.

```
# 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
```

Langkah 2: Membuat dan men-deploy lingkungan zappa

Pada langkah ini Anda akan menggunakan Zappa untuk secara otomatis mengonfigurasi titik akhir API Gateway dan kemudian men-deploy ke Lambda.

1. Inisialisasi Zappa dari dalam `serverless_application` direktori. Sebagai contoh, kami menggunakan pengaturan default, tetapi jika Anda memiliki preferensi kustomisasi, Zappa menampilkan petunjuk konfigurasi.

```
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. Aktifkan X-Ray. Buka file `zappa_settings.json` dan memverifikasi bahwa itu terlihat mirip dengan contoh.

```
{
```

```

    "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. Tambahkan `"xray_tracing": true` sebagai entri ke file konfigurasi.

```

{
  "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. Men-deploy aplikasi. Ini secara otomatis mengonfigurasi titik akhir API Gateway dan mengunggah kode Anda ke Lambda.

```
zappa deploy
```

```

...
Deploying API Gateway..
Deployment complete!: https://*****.execute-api.us-west-2.amazonaws.com/dev

```

Langkah 3: Aktifkan penelusuran X-Ray untuk API Gateway

Pada langkah ini Anda akan berinteraksi dengan konsol API Gateway untuk mengaktifkan pelacakan X-Ray.

1. Masuk ke AWS Management Console dan buka konsol API Gateway di <https://console.aws.amazon.com/apigateway/>.

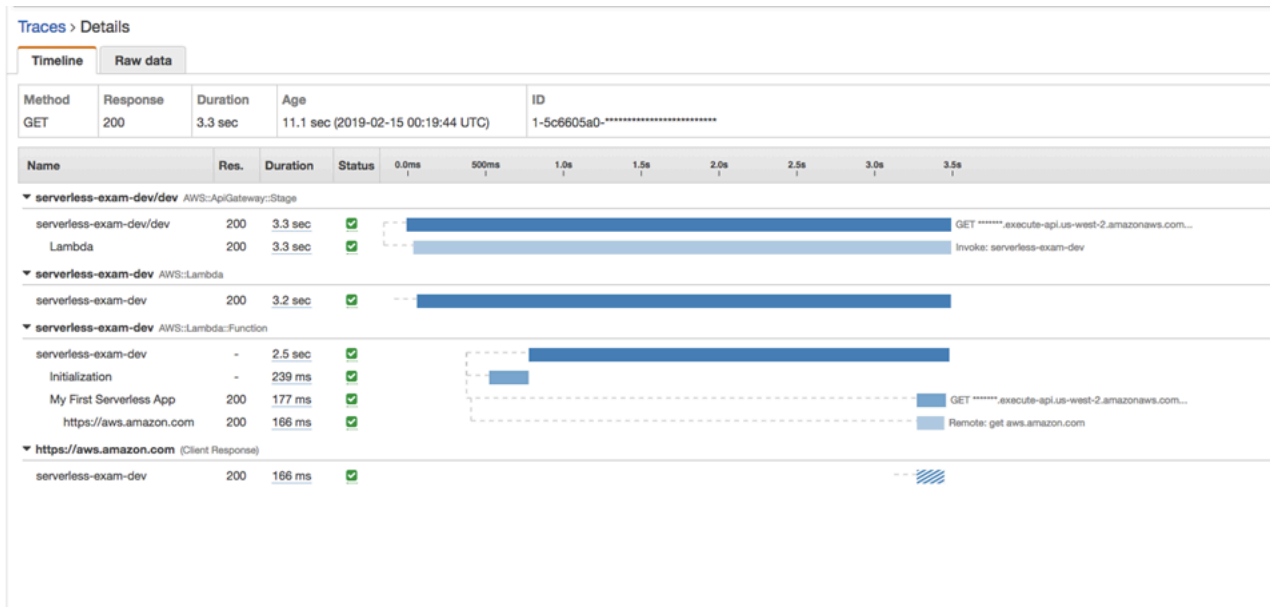
2. Temukan API yang baru dibuat. Harus terlihat seperti `serverless-exam-dev`.
3. Memilih Tahapan.
4. Pilih nama tahap deployment Anda. Default-nya adalah `dev`.
5. Pada tab Masuk/Penelusuran, pilih Aktifkan penelusuran X-Ray.
6. Pilih Simpan Perubahan.
7. Akses titik akhir di peramban Anda. Jika Anda menggunakan contoh aplikasi Hello World, aplikasi tersebut harus menampilkan hal berikut.

```
"Hello, World: https://aws.amazon.com/"
```

Langkah 4: Lihat telusuran yang dibuat

Pada langkah ini Anda akan berinteraksi dengan konsol X-Ray untuk melihat telusuran pelacakan yang dibuat oleh aplikasi contoh. Untuk walkthrough yang lebih detail tentang analisis penelusuran, lihat [Melihat Peta Layanan](#).

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Lihat segmen yang dihasilkan oleh API Gateway, fungsi Lambda, dan kontainer Lambda.
3. Di bawah segmen fungsi Lambda, melihat subsegmen bernama `My First Serverless App`. Ini diikuti oleh subsegmen kedua bernama `https://aws.amazon.com`.
4. Selama inisialisasi, Lambda mungkin juga menghasilkan subsegmen ketiga bernama `initialization`.



Langkah 5: Membersihkan

Selalu akhiri sumber daya yang tidak lagi Anda gunakan untuk menghindari akumulasi biaya tak terduga. Seperti tutorial ini menunjukkan, alat-alat seperti Zappa merampingkan redeployment nserver.

Untuk menghapus aplikasi Anda dari Lambda, API Gateway, dan Amazon S3, jalankan perintah berikut di direktori proyek Anda dengan menggunakan AWS CLI.


```
zappa undeploy dev
```

Langkah selanjutnya

Tambahkan lebih banyak fitur ke aplikasi Anda dengan menambahkan AWS klien dan menginstrumentasinya dengan X-Ray. Pelajari selengkapnya tentang opsi komputasi nserver di [Nirserver AWS](#).

Bekerja dengan .NET

Ada dua cara untuk instrumen aplikasi .NET Anda untuk mengirim jejak ke X-Ray:

- [AWS Distro OpenTelemetry untuk .NET — AWS Distribusi yang menyediakan serangkaian pustaka open source untuk mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan termasuk Amazon CloudWatch,, AWS X-Ray dan Amazon OpenSearch Service, melalui Distro for Collector.AWS OpenTelemetry](#)
- [AWS X-Ray SDK for .NET — Satu set perpustakaan untuk menghasilkan dan mengirim jejak ke X-Ray melalui daemon X-Ray.](#)

Untuk informasi selengkapnya, lihat [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#).

AWS Distro para OpenTelemetry .NET

Dengan AWS Distro OpenTelemetry untuk .NET, Anda dapat menginstruksikan aplikasi Anda sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa solusi AWS pemantauan termasuk Amazon CloudWatch,, AWS X-Ray dan Amazon Service. OpenSearch Menggunakan X-Ray dengan AWS Distro untuk OpenTelemetry membutuhkan dua komponen: OpenTelemetry SDK diaktifkan untuk digunakan dengan X-Ray, dan AWS Distro untuk OpenTelemetry Kolektor diaktifkan untuk digunakan dengan X-Ray.

Untuk memulai, lihat [dokumentasi AWS Distro for OpenTelemetry .NET](#).

Untuk informasi selengkapnya tentang menggunakan AWS Distro untuk OpenTelemetry with AWS X-Ray dan lainnya Layanan AWS, lihat [AWS Distro untuk OpenTelemetry atau Distro untuk AWS Dokumentasi](#). OpenTelemetry

Untuk informasi selengkapnya tentang dukungan dan penggunaan bahasa, lihat [AWS Observabilitas di GitHub](#).

AWS X-Ray SDK for .NET

X-Ray SDK untuk .NET adalah perpustakaan untuk menginstrumentasi aplikasi web C # .NET, aplikasi web .NET Core, dan fungsi-fungsi .NET Core. AWS Lambda Ini menyediakan kelas dan metode untuk menghasilkan dan mengirim penelusuran data ke [Daemon X-Ray](#). Ini termasuk

informasi tentang permintaan masuk yang dilayani oleh aplikasi, dan panggilan yang dilakukan aplikasi ke hilir Layanan AWS, web HTTP APIs, dan database SQL.

Note

X-Ray SDK for .NET merupakan proyek sumber terbuka. Anda dapat mengikuti proyek dan mengirimkan masalah dan menarik permintaan di GitHub: github.com/aws/aws-xray-sdk-dotnet

Untuk aplikasi web, mulai dengan [menambahkan penanganan pesan ke konfigurasi web Anda](#) untuk melacak permintaan yang masuk. Penanganan pesan membuat [segmen](#) untuk setiap permintaan yang ditelusuri, dan melengkapi segmen ketika respons dikirim. Ketika segmen terbuka Anda dapat menggunakan metode klien SDK untuk menambahkan informasi ke segmen dan membuat subsegmen untuk pelacakan panggilan hilir. SDK juga secara otomatis mencatat pengecualian yang aplikasi Anda lempar ketika segmen terbuka.

Untuk fungsi Lambda disebut oleh instrumen aplikasi atau layanan, Lambda membaca [tracing header](#) dan pelacakan sampel permintaan secara otomatis. Untuk fungsi lainnya, Anda dapat [mengonfigurasi Lambda](#) untuk sampel dan pelacakan permintaan masuk. Dalam kedua kasus, Lambda membuat segmen dan menyediakannya ke X-Ray SDK.

Note

Pada Lambda, X-Ray SDK adalah opsional. Jika Anda tidak menggunakannya dalam fungsi Anda, peta layanan Anda masih akan menyertakan simpul untuk layanan Lambda, dan satu untuk setiap fungsi Lambda. Dengan menambahkan SDK, Anda dapat melakukan instrumen kode fungsi Anda untuk menambahkan subsegmen ke segmen fungsi yang dicatat oleh Lambda. Lihat [AWS Lambda dan AWS X-Ray](#) untuk informasi selengkapnya.

Berikutnya, gunakan X-Ray SDK for .NET untuk [Instrumen klien AWS SDK untuk .NET Anda](#).

Setiap kali Anda melakukan panggilan ke hilir Layanan AWS atau sumber daya dengan klien yang diinstrumentasi, SDK akan mencatat informasi tentang panggilan di subsegmen. AWS layanan dan sumber daya yang Anda akses dalam layanan muncul sebagai node hilir pada peta jejak untuk membantu Anda mengidentifikasi kesalahan dan masalah pembatasan pada koneksi individual.

[X-Ray SDK for .NET juga menyediakan instrumentasi untuk panggilan hilir ke database APIs web HTTP dan SQL](#). Metode ekstensi `GetResponseTraced` untuk panggilan HTTP keluar penelusuran

`System.Net.HttpWebRequest`. Anda dapat menggunakan versi X-Ray SDK untuk .NET dari `SqlCommand` untuk melakukan instrumen kueri SQL.

Setelah Anda mulai menggunakan SDK, sesuaikan perilakunya dengan [mengonfigurasi perekam dan penanganan pesan](#). Anda dapat menambahkan plugin untuk mencatat data mengenai sumber daya komputasi yang berjalan di aplikasi Anda, menyesuaikan perilaku sampling dengan mendefinisikan aturan sampling, dan mengatur tingkat log untuk melihat lebih atau kurang informasi dari SDK dalam log aplikasi Anda.

Catat informasi tambahan tentang permintaan dan pekerjaan yang dilakukan aplikasi Anda dalam [anotasi dan metadata](#). Anotasi adalah pasangan kunci-nilai sederhana yang diindeks untuk digunakan dengan [ekspresi filter](#), sehingga Anda dapat mencari pelacakan yang berisi data tertentu. Entri metadata kurang bersifat membatasi dan dapat mencatat seluruh objek dan array — segala yang dapat disambungkan ke dalam JSON.

Anotasi dan Metadata

Anotasi dan metadata adalah teks abritari yang Anda tambahkan ke segmen dengan X-Ray SDK. Anotasi diindekskan untuk digunakan dengan ekspresi filter. Metadata tidak diindeks, tetapi dapat dilihat di segmen mentah dengan konsol X-Ray atau API. Siapa pun yang Anda berikan akses baca ke X-Ray dapat melihat data ini.

Bila Anda memiliki banyak klien diinstrumentasi dalam kode Anda, segmen permintaan tunggal dapat berisi sejumlah besar subsegmen, satu untuk setiap panggilan yang dibuat dengan klien berinstrumen. Anda dapat mengatur dan mengelompokkan subsegmen dengan membungkus panggilan klien di [subsegmen kustom](#). Anda dapat membuat subsegmen kustom untuk seluruh fungsi atau bagian dari kode, dan mencatat metadata dan anotasi pada subsegmen bukan menulis segala sesuatu pada segmen induk.

Untuk referensi dokumentasi tentang SDK kelas dan metode, lihat hal berikut:

- [AWS X-Ray SDK for .NET API Referensi](#)
- [AWS X-Ray SDK for .NET Core API Referensi](#)

Paket yang sama mendukung baik NET dan NET Inti, tetapi kelas yang digunakan bervariasi. Contoh dalam link Bab ini ke referensi API .NET kecuali kelas khusus untuk Inti .NET.

Persyaratan

X-Ray SDK for .NET membutuhkan .NET Framework 4.5 atau yang lebih baru AWS SDK untuk .NET dan.

Untuk aplikasi NET Core dan fungsi, SDK membutuhkan NET Core 2.0 atau yang lebih baru.

Menambahkan X-Ray SDK for .NET untuk aplikasi Anda

Gunakan NuGet untuk menambahkan X-Ray SDK for .NET ke aplikasi Anda.

Untuk menginstal X-Ray SDK untuk.NET NuGet dengan manajer paket di Visual Studio

1. Pilih Tools, NuGet Package Manager, Manage NuGet Packages for Solution.
2. Cari AWSXRayPerekam.
3. Pilih paket, dan kemudian pilih Instal.

Manajemen dependensi

X-Ray SDK for .NET tersedia dari [Nuget](#). Instal SDK menggunakan manajer paket:

```
Install-Package AWSXRayRecorder -Version 2.10.1
```

Paket nuget AWSXRayRecorder v2.10.1 memiliki dependensi berikut:

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)
```

Untuk detail selengkapnya tentang manajemen dependensi, lihat dokumentasi Microsoft tentang [Ketergantungan nuget](#) dan [Resolusi dependensi nuget](#).

Mengonfigurasi X-Ray SDK for .NET

Anda dapat mengonfigurasi X-Ray SDK for .NET dengan plugin untuk menyertakan informasi tentang layanan tempat aplikasi Anda berjalan, mengubah perilaku pengambilan sampel default, atau menambahkan aturan pengambilan sampel yang berlaku untuk permintaan ke jalur tertentu.

Untuk aplikasi web .NET, tambahkan kunci ke bagian appSettings file Web.config Anda.

Example Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Untuk .NET Core, buat file bernama appsettings.json dengan kunci tingkat atas bernama XRay.

Example .NET appsettings.json

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin",
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

Lalu, dalam kode aplikasi Anda, bangun objek konfigurasi dan gunakan untuk menginisialisasi pencatat X-Ray. Lakukan ini sebelum Anda [menginisialisasi pencatat](#).

Example .NET Core Program.cs – Konfigurasi pencatat

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Jika Anda menginstrumentasikan aplikasi web .NET Core, Anda juga dapat meneruskan objek konfigurasi ke metode UseXRay saat Anda [mengonfigurasi pengendali pesan](#). Untuk fungsi Lambda, gunakan metode InitializeInstance seperti yang ditunjukkan di atas.

Untuk informasi selengkapnya tentang API konfigurasi .NET Core, lihat [Konfigurasi aplikasi ASP.NET Core](#) di docs.microsoft.com.

Bagian-bagian

- [Plugin](#)
- [Aturan pengambilan sampel](#)
- [Pencatatan \(NET\)](#)
- [Pencatatan \(.NET Core\)](#)
- [Variabel-variabel lingkungan](#)

Plugin

Gunakan plugin untuk menambahkan data tentang layanan yang menjadi meng-hosting aplikasi Anda.

Plugin

- Amazon EC2 — EC2Plugin menambahkan ID instans, Availability Zone, dan Grup CloudWatch Log.
- Elastic Beanstalk – ElasticBeanstalkPlugin menambahkan nama lingkungan, label versi, dan ID deployment.
- Amazon ECS – ECSPlugin menambahkan ID kontainer.

Untuk menggunakan plugin, konfigurasi klien X-Ray SDK for .NET dengan menambahkan pengaturan `AWSXRayPlugins`. Jika beberapa plugin berlaku untuk aplikasi Anda, tentukan semuanya dalam pengaturan yang sama, dipisahkan dengan koma.

Example Web.config - plugin

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin,ElasticBeanstalkPlugin"/>
  </appSettings>
</configuration>
```


Example .NET Core appsettings.json – Plugin

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin,ElasticBeanstalkPlugin"
  }
}
```

Aturan pengambilan sampel

SDK menggunakan aturan pengambilan sampel yang Anda tetapkan di konsol X-Ray untuk menentukan permintaan yang akan dicatat. Aturan default menelusuri permintaan pertama setiap detik, dan lima persen permintaan tambahan di semua layanan yang mengirim pelacakan ke X-Ray. [Buat aturan tambahan di konsol X-Ray](#) untuk menyesuaikan jumlah data yang dicatat untuk setiap aplikasi Anda.

SDK menerapkan aturan kustom sesuai urutan penetapannya. Jika permintaan cocok dengan beberapa aturan kustom, SDK hanya menerapkan aturan pertama.

Note

Jika SDK tidak dapat mencapai X-Ray untuk mendapatkan aturan pengambilan sampel, SDK akan beralih ke aturan lokal default dari permintaan pertama setiap detik, dan lima persen permintaan tambahan per host. Hal ini dapat terjadi jika host tidak memiliki izin untuk memanggil sampling APIs, atau tidak dapat terhubung ke daemon X-Ray, yang bertindak sebagai proxy TCP untuk panggilan API yang dilakukan oleh SDK.

Anda juga dapat mengonfigurasi SDK untuk memuat aturan sampling dari dokumen JSON. SDK dapat menggunakan aturan lokal sebagai cadangan jika terjadi kasus tidak dapat mengambil sampel X-Ray, atau menggunakan aturan lokal secara eksklusif.

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
}
}
```

Contoh ini menentukan satu aturan kustom dan aturan default. Aturan kustom menerapkan tingkat pengambilan sampel lima persen tanpa jumlah minimum permintaan untuk melacak jalur di `/api/move/`. Aturan default menelusuri permintaan pertama setiap detik dan 10 persen dari permintaan tambahan.

Kerugian dari menentukan aturan secara lokal adalah bahwa target tetap diterapkan oleh setiap instans pencatat secara independen, alih-alih dikelola oleh layanan X-Ray. Ketika Anda men-deploy lebih banyak host, laju tetap akan dikalikan, sehingga sulit untuk mengontrol jumlah data yang dicatat.

Pada AWS Lambda, Anda tidak dapat mengubah laju pengambilan sampel. Jika fungsi Anda dipanggil oleh layanan yang diinstrumentasikan, panggilan yang menghasilkan permintaan yang sampelnya diambil oleh layanan yang akan dicatat oleh Lambda. Jika pelacakan aktif diaktifkan dan tidak ada header pelacakan, Lambda membuat keputusan pengambilan sampel.

Untuk mengonfigurasi aturan pencadangan, beri tahu X-Ray SDK for .NET agar memuat aturan pengambilan sampel dari file dengan pengaturan `SamplingRuleManifest` tersebut.

Example .NET Web.config - aturan pengambilan sampel

```
<configuration>
  <appSettings>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Example .NET Core appsettings.json – Aturan Pengambilan Sampel

```
{
```

```
"XRay": {  
  "SamplingRuleManifest": "sampling-rules.json"  
}
```

Untuk hanya menggunakan aturan lokal, buat pencatat dengan `LocalizedSamplingStrategy`. Jika Anda memiliki aturan pencadangan yang dikonfigurasi, hapus konfigurasi tersebut.

Example .NET global.asax – Aturan pengambilan sampel lokal

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new  
    LocalizedSamplingStrategy("samplingrules.json")).Build();  
AWSXRayRecorder.InitializeInstance(recorder: recorder);
```

Example .NET Core Program.cs – Aturan pengambilan sampel lokal

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new  
    LocalizedSamplingStrategy("sampling-rules.json")).Build();  
AWSXRayRecorder.InitializeInstance(configuration, recorder);
```

Pencatatan (NET)

X-Ray SDK for .NET menggunakan mekanisme pencatatan yang sama dengan [AWS SDK untuk .NET](#). Jika Anda sudah mengonfigurasi aplikasi untuk mencatat AWS SDK untuk .NET output, konfigurasi yang sama berlaku untuk output dari X-Ray SDK for .NET.

Untuk mengonfigurasi pencatatan, tambahkan bagian konfigurasi bernama `aws` ke file `App.config` Anda atau file `Web.config`.

Example Web.config - pencatatan

```
...  
<configuration>  
  <configSections>  
    <section name="aws" type="Amazon.AWSSection, AWSSDK.Core"/>  
  </configSections>  
  <aws>  
    <logging logTo="Log4Net"/>  
  </aws>  
</configuration>
```

Untuk informasi selengkapnya, lihat [Mengonfigurasi Aplikasi AWS SDK untuk .NET Anda](#) dalam Panduan Developer AWS SDK untuk .NET .

Pencatatan (.NET Core)

X-Ray SDK for .NET menggunakan opsi pencatatan yang sama dengan [AWS SDK untuk .NET](#). Untuk mengonfigurasi pencatatan untuk aplikasi .NET Core, teruskan opsi pencatatan ke metode `AWSSXRayRecorder.RegisterLogger`.

Misalnya, untuk menggunakan log4net, buat file konfigurasi yang menentukan logger, format output, dan lokasi 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>
```

Lalu, buat logger dan terapkan konfigurasi di kode program Anda.

Example .NET Core Program.cs – Pencatatan

```
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)
{
    ...
}
```

Untuk informasi selengkapnya tentang mengonfigurasi log4net, lihat [Konfigurasi](#) di logging.apache.org.

Variabel-variabel lingkungan

Anda dapat menggunakan variabel lingkungan untuk mengonfigurasi X-Ray SDK for .NET. SDK mendukung variabel berikut.

- `AWS_XRAY_TRACING_NAME` – Menetapkan nama layanan yang digunakan SDK untuk segmen. Menimpa nama layanan yang Anda tetapkan pada [strategi penamaan segmen](#) filter servlet.
- `AWS_XRAY_DAEMON_ADDRESS` – Mengatur host dan port listener daemon X-Ray. Secara default, SDK menggunakan `127.0.0.1:2000` untuk data pelacakan (UDP) dan pengambilan sampel (TCP). Gunakan variabel ini jika Anda telah mengonfigurasi daemon untuk [mendengarkan di port berbeda](#) atau jika berjalan pada host yang berbeda.

format

- Port yang sama – *address:port*
- Port yang berbeda – `tcp:address:port` `udp:address:port`
- `AWS_XRAY_CONTEXT_MISSING`— Setel `RUNTIME_ERROR` untuk melempar pengecualian saat kode instrumentasi Anda mencoba merekam data saat tidak ada segmen yang terbuka.

Nilai Valid

- `RUNTIME_ERROR`— Lempar pengecualian runtime.
- `LOG_ERROR`— Log kesalahan dan lanjutkan (default).
- `IGNORE_ERROR`— Abaikan kesalahan dan lanjutkan.

Kesalahan yang terkait dengan segmen atau subsegmen yang hilang dapat terjadi saat Anda mencoba menggunakan klien yang diinstrumentasikan dalam kode startup yang berjalan saat tidak ada permintaan yang dibuka, atau dalam kode yang memunculkan utas baru.

Menginstrumentasikan permintaan HTTP yang masuk dengan X-Ray SDK for .NET

Anda dapat menggunakan X-Ray SDK untuk melacak permintaan HTTP masuk yang dilayani aplikasi Anda pada EC2 instans di Amazon EC2 AWS Elastic Beanstalk, atau Amazon ECS.

Gunakan pengendali pesan untuk menginstrumentasi permintaan HTTP yang masuk. Saat Anda menambahkan pengendali pesan X-Ray ke aplikasi Anda, X-Ray SDK for .NET membuat segmen untuk setiap permintaan yang dibuat sampel. Segmen ini meliputi waktu, metode, dan disposisi permintaan HTTP. Instrumentasi tambahan membuat subsegmen pada segmen ini.

Note

Untuk AWS Lambda fungsi, Lambda membuat segmen untuk setiap permintaan sampel. Untuk informasi selengkapnya, lihat [AWS Lambda dan AWS X-Ray](#).

Setiap segmen memiliki nama yang mengidentifikasi aplikasi Anda dalam peta layanan. Segmen dapat diberi nama secara statis, atau Anda dapat mengonfigurasi SDK untuk nama itu secara dinamis berdasarkan header host dalam permintaan masuk. Penamaan dinamis memungkinkan Anda mengelompokkan pelacakan berdasarkan nama domain dalam permintaan, dan menerapkan nama default jika nama tersebut tidak cocok dengan pola yang diharapkan (misalnya, jika header host ditiru).

Permintaan yang Diteruskan

Jika penyeimbang beban atau perantara lainnya meneruskan permintaan ke aplikasi Anda, X-Ray akan mengambil IP klien dari header X-Forwarded-For dalam permintaan bukan dari sumber IP dalam paket IP. IP klien yang tercatat untuk permintaan yang diteruskan dapat dipalsukan, jadi tidak boleh dipercaya.

Pengendali pesan membuat segmen untuk setiap permintaan yang masuk dengan blok `http` yang berisi informasi berikut:

- Metode HTTP – DAPATKAN, POSTING, LETAKKAN, HAPUS, dll.
- Alamat klien – Alamat IP klien yang mengirim permintaan.

- Kode respons – Kode respons HTTP untuk permintaan yang selesai.
- Timing – Waktu mulai (saat permintaan diterima) dan waktu akhir (saat respons dikirim).
- Agen pengguna — user-agent dari permintaan.
- Panjang konten — content-length dari respons.

Bagian-bagian

- [Menginstrumentasikan permintaan masuk \(.NET\)](#)
- [Menginstrumentasikan permintaan yang masuk \(.NET Core\)](#)
- [Mengonfigurasi strategi penamaan segmen](#)

Menginstrumentasikan permintaan masuk (.NET)

Untuk menginstrumentasi permintaan yang dilayani oleh aplikasi Anda, panggil `RegisterXRay` di dalam metode `Init` dari file `global.asax` Anda.

Example `global.asax` - pengendali pesan

```
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");
        }
    }
}
```

Menginstrumentasikan permintaan yang masuk (.NET Core)

Untuk menginstrumentasi permintaan yang dilayani oleh aplikasi Anda, panggil metode `UseXRay` sebelum middleware lain dalam metode `Configure` kelas `Startup` Anda, karena middleware X-Ray idealnya harus menjadi middleware pertama yang memproses permintaan dan middleware terakhir yang memproses respons dalam alur.

Note

.NET Core 2.0, jika Anda memiliki metode `UseExceptionHandler` dalam aplikasi, pastikan untuk memanggil `UseXRay` setelah metode `UseExceptionHandler` guna memastikan bahwa pengecualian tercatat.

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
    ...
}
```

Metode `UseXRay` juga dapat mengambil [objek konfigurasi](#) sebagai argumen kedua.

```
app.UseXRay("MyApp", configuration);
```

Mengonfigurasi strategi penamaan segmen

AWS X-Ray menggunakan nama layanan untuk mengidentifikasi aplikasi Anda dan membedakannya dari aplikasi lain, database, eksternal APIs, dan AWS sumber daya yang digunakan aplikasi Anda.

Saat SDK X-Ray membuat segmen untuk permintaan masuk, SDK akan mencatat nama layanan aplikasi Anda di [kolom nama](#).

SDK X-Ray dapat memberi nama segmen setelah nama host di header permintaan HTTP. Namun, header ini dapat ditiru, yang dapat mengakibatkan simpul tak terduga di peta layanan Anda. Untuk mencegah SDK dari penamaan segmen salah karena permintaan dengan header host palsu, Anda harus menentukan nama default untuk permintaan masuk.

Jika aplikasi Anda menyuguhkan permintaan untuk beberapa domain, Anda dapat mengonfigurasi SDK untuk menggunakan strategi penamaan dinamis untuk mencerminkan ini dalam nama segmen. Strategi penamaan dinamis mengizinkan SDK menggunakan nama host untuk permintaan yang sesuai dengan pola yang diharapkan, dan menerapkan nama default untuk permintaan yang tidak sesuai.

Misalnya, Anda boleh memiliki satu aplikasi yang melayani permintaan untuk tiga subdomain—`www.example.com`, `api.example.com`, dan `static.example.com`. Anda dapat menggunakan strategi penamaan dinamis dengan pola `*.example.com` untuk mengidentifikasi segmen untuk setiap subdomain dengan nama yang berbeda, mengakibatkan tiga simpul layanan pada peta layanan. Jika aplikasi Anda menerima permintaan dengan nama host yang tidak cocok dengan polanya, Anda akan melihat simpul keempat di peta layanan dengan nama cadangan yang Anda tentukan.

Untuk menggunakan nama yang sama untuk semua segmen permintaan, tentukan nama aplikasi Anda saat menginisialisasi pengendali pesan, seperti yang ditunjukkan di [bagian sebelumnya](#). Tindakan ini memiliki efek yang sama seperti membuat [FixedSegmentNamingStrategy](#) dan meneruskannya ke metode `RegisterXRay`.

```
AWSXRayAspNet.RegisterXRay(this, new FixedSegmentNamingStrategy("MyApp"));
```

Note

Anda dapat mengganti nama layanan default yang Anda tentukan dalam kode dengan [variabel lingkungan](#) `AWS_XRAY_TRACING_NAME`.

Strategi penamaan dinamis menentukan pola yang harus cocok dengan nama host, dan nama default yang digunakan jika nama host dalam permintaan HTTP tidak cocok dengan pola tersebut. Untuk memberi nama segmen secara dinamis, buat [DynamicSegmentNamingStrategy](#) dan teruskan ke metode `RegisterXRay`.

```
AWSXRayAspNet.RegisterXRay(this, new DynamicSegmentNamingStrategy("MyApp",
    "*.example.com"));
```

Menelusuri panggilan AWS SDK dengan X-Ray SDK untuk .NET

[Saat aplikasi Anda melakukan panggilan Layanan AWS untuk menyimpan data, menulis ke antrian, atau mengirim notifikasi, X-Ray SDK for .NET melacak panggilan hilir di subsegmen.](#) Ditelusuri Layanan AWS dan sumber daya yang Anda akses dalam layanan tersebut (misalnya, bucket Amazon S3 atau antrian Amazon SQS), muncul sebagai node hilir pada peta jejak di konsol X-Ray.

Anda dapat menginstruksikan semua AWS SDK untuk .NET klien Anda dengan menelepon `RegisterXRayForAllServices` sebelum Anda membuatnya.

Example `SampleController.cs` - instrumentasi klien DynamoDB

```
using Amazon;
using Amazon.Util;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.AwsSdk;

namespace SampleEBWebApplication.Controllers
{
    public class SampleController : ApiController
    {
        AWS SDK Handler.RegisterXRayForAllServices();
        private static readonly Lazy<AmazonDynamoDBClient> LazyDdbClient = new
        Lazy<AmazonDynamoDBClient>(() =>
        {
            var client = new AmazonDynamoDBClient(EC2InstanceMetadata.Region ??
            RegionEndpoint.USEast1);
            return client;
        });
    }
}
```

Untuk instrumen klien agar beberapa layanan dan bukan yang lain, hubungi `RegisterXRay` sebagai gantinya `RegisterXRayForAllServices`. Mengganti teks yang disorot dengan nama antarmuka klien layanan.

```
AWS SDK Handler.RegisterXRay<IAmazonDynamoDB>()
```

Untuk semua layanan, Anda dapat melihat nama API yang disebut dalam konsol X-Ray. Untuk subset layanan, X-Ray SDK menambahkan informasi ke segmen untuk memberikan lebih banyak perincian di peta layanan.

Sebagai contoh, ketika Anda melakukan panggilan dengan klien DynamoDB berinstrumen, SDK menambahkan nama tabel ke segmen untuk panggilan yang menargetkan tabel. Di konsol tersebut, setiap tabel muncul sebagai simpul terpisah di peta layanan, dengan simpul DynamoDB generik untuk panggilan yang tidak menargetkan tabel.

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Ketika Anda mengakses sumber daya bernama, panggilan ke layanan berikut membuat simpul tambahan di peta layanan. Panggilan yang tidak menargetkan sumber daya tertentu membuat simpul generik untuk layanan tersebut.

- Amazon DynamoDB – Nama tabel
- Amazon Simple Storage Service – Bucket dan nama kunci
- Amazon Simple Queue Service – Nama antrean

Pelacakan panggilan ke layanan web HTTP hilir dengan X-Ray SDK for .NET

Ketika aplikasi Anda melakukan panggilan ke microservices atau HTTP publik APIs, Anda dapat menggunakan X-Ray SDK untuk metode `GetResponseTraced` ekstensi `System.Net.HttpWebRequest` untuk instrumen panggilan tersebut dan menambahkan API ke grafik layanan sebagai layanan hilir.

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();
}
```

Untuk panggilan asinkron, gunakan `GetAsyncResponseTraced`.

```
request.GetAsyncResponseTraced();
```

Jika Anda menggunakan [system.net.http.httpclient](#), gunakan pengendali penyerahan `HttpClientXRayTracingHandler` untuk mencatat panggilan.

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);
}
```

Ketika Anda menginstrumen panggilan ke API web hilir, X-Ray SDK for .NET mencatat subsegmen dengan informasi tentang permintaan dan respons HTTP. X-Ray menggunakan subsegmen untuk membuat segmen yang disimpulkan untuk API.

Example Subsegmen untuk panggilan HTTP hilir

```
{
  "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 Segmen yang disimpulkan untuk panggilan HTTP downstream

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
}
```

```
"inferred": true
}
```

Pelacakan kueri SQL dengan X-Ray SDK for .NET

X-Ray SDK for .NET menyediakan kelas pembungkus untuk `System.Data.SqlClient.SqlCommand`, bernama `TraceableSqlCommand`, yang dapat Anda gunakan sebagai pengganti `SqlCommand`. Anda dapat menginisialisasi perintah SQL dengan kelas `TraceableSqlCommand`.

Pelacakan kueri SQL dengan metode sinkron dan asinkron

Contoh berikut menunjukkan tentang menggunakan `TraceableSqlCommand` untuk secara otomatis melacak kueri SQL Server secara sinkron dan asinkron.

Example **Controller.cs** - Instrumentasi klien SQL (sinkron)

```
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();
    }
}
```

Anda dapat menjalankan kueri secara asinkron dengan menggunakan metode `ExecuteReaderAsync`.

Example **Controller.cs** - Instrumentasi klien SQL (asinkron)

```
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();
    }
}
```

Mengumpulkan kueri SQL yang dibuat untuk SQL Server

Anda dapat mengaktifkan penangkapan `SqlCommand.CommandText` sebagai bagian dari subsegment yang dibuat oleh kueri SQL Anda. `SqlCommand.CommandText` muncul sebagai bidang `sanitized_query` di subsegmen JSON. Secara default, fitur ini dinonaktifkan untuk keamanan.

Note

Jangan mengaktifkan fitur pengumpulan jika Anda menyertakan informasi sensitif sebagai teks yang jelas dalam kueri SQL Anda.

Anda dapat mengaktifkan pengumpulan kueri SQL dengan dua cara:

- Atur properti `CollectSqlQueries` untuk `true` dalam konfigurasi global untuk aplikasi Anda.
- Atur parameter `collectSqlQueries` dalam instans `TraceableSqlCommand` untuk `true` mengumpulkan panggilan dalam instans.

Aktifkan `CollectSqlQueries` properti global

Contoh berikut menunjukkan tentang mengaktifkan properti `CollectSqlQueries` untuk NET dan Inti .NET.

.NET

Untuk mengatur properti `CollectSqlQueries` untuk `true` dalam konfigurasi global dari aplikasi Anda di .NET, modifikasi `appsettings` dari file `App.config` atau `Web.config` Anda, seperti yang ditunjukkan.

Example **App.config** Atau **Web.config** – Aktifkan pengumpulan Kueri SQL secara global

```
<configuration>
<appSettings>
  <add key="CollectSqlQueries" value="true">
</appSettings>
</configuration>
```

.NET Core

Untuk mengatur properti `CollectSqlQueries` untuk `true` dalam konfigurasi global dari aplikasi Anda di .NET Core, modifikasi file `appsettings.json` Anda di bawah kunci `X-Ray`, seperti yang ditunjukkan.

Example **appsettings.json** – Aktifkan pengumpulan Kueri SQL secara global

```
{
  "XRay": {
    "CollectSqlQueries": "true"
  }
}
```

Aktifkan `collectSqlQueries` parameter

Anda dapat mengatur parameter `collectSqlQueries` dalam instans `TraceableSqlCommand` ke `true` untuk mengumpulkan teks kueri SQL untuk kueri SQL Server dibuat menggunakan instans tersebut. Mengatur parameter ke `false` menonaktifkan fitur `CollectSqlQuery` untuk instans `TraceableSqlCommand`.

Note

Nilai `collectSqlQueries` dalam instans `TraceableSqlCommand` mengganti nilai yang ditetapkan dalam konfigurasi global properti `CollectSqlQueries`.

Example Contoh **Controller.cs** – Aktifkan pengumpulan Kueri SQL untuk instans

```
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();  
        }  
}
```

Membuat subsegmen tambahan

Subsegmen memperluas [segmen](#) pelacakan dengan detail tentang pekerjaan yang dilakukan untuk melayani permintaan. Setiap kali Anda melakukan panggilan dengan klien berinstrumen, X-Ray tersebut mencatat informasi yang dihasilkan dalam subsegmen. Anda dapat membuat subsegment tambahan untuk mengelompokkan subsegment lain, untuk mengukur performa bagian kode, atau untuk mencatat anotasi dan metadata.

Untuk mengelola subsegment, gunakan metode `BeginSubsegment` dan `EndSubsegment`. Lakukan pekerjaan apa pun di subsegmen dalam satu blok `try` dan gunakan `AddException` untuk melacak pengecualian. Panggilan `EndSubsegment` dalam blok `finally` untuk memastikan bahwa subsegmen ditutup.

Example Controller.cs – subsegmen Kustom

```
AWSXRayRecorder.Instance.BeginSubsegment("custom method");  
try  
{  
    DoWork();  
}  
catch (Exception e)  
{  
    AWSXRayRecorder.Instance.AddException(e);  
}  
finally  
{  
    AWSXRayRecorder.Instance.EndSubsegment();  
}
```

Ketika Anda membuat subsegmen dalam segmen atau subsegmen lain, X-Ray SDK for .NET menghasilkan ID untuk subsegmen dan mencatat waktu mulai dan waktu berakhir.

Example Subsegmen dengan metadata

```
"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"
    }
  },
},
```

Tambahkan anotasi dan metadata ke segmen dengan X-Ray SDK for .NET

Anda dapat mencatat informasi tambahan tentang permintaan, lingkungan, atau aplikasi Anda dengan anotasi dan metadata. Anda dapat menambahkan anotasi dan metadata ke segmen yang dibuat oleh SDK X-Ray, atau subsegmen kustom yang Anda buat.

Anotasi adalah pasangan kunci-nilai dengan string, nomor, atau nilai-nilai Boolean. Anotasi diindekskan untuk digunakan dengan [Ekspresi filter](#). Gunakan anotasi untuk mencatat data yang ingin Anda gunakan untuk mengelompokkan pelacakan di konsol tersebut, atau saat memanggil API [GetTraceSummaries](#).

Metadata adalah pasangan kunci-nilai yang dapat memiliki nilai dari setiap tipe, termasuk objek dan daftar, tetapi tidak diindekskan untuk digunakan dengan ekspresi filter. Gunakan metadata untuk mencatat data tambahan yang ingin Anda simpan di pelacakan tetapi tidak perlu digunakan dengan pencarian.

Bagian-bagian

- [Mencatat anotasi dengan X-Ray SDK for .NET](#)
- [Mencatat metadata dengan X-Ray SDK for .NET](#)

Mencatat anotasi dengan X-Ray SDK for .NET

Gunakan anotasi untuk mencatat informasi pada segmen atau subsegmen yang ingin Anda indeks untuk pencarian.

Berikut ini diperlukan untuk semua anotasi dalam X-Ray:

Persyaratan Anotasi

- Tombol — Kunci untuk anotasi X-Ray dapat memiliki hingga 500 karakter alfanumerik. Anda tidak dapat menggunakan spasi atau simbol selain titik atau titik (.)
- Nilai — Nilai untuk anotasi X-Ray dapat memiliki hingga 1.000 karakter Unicode.
- Jumlah Anotasi — Anda dapat menggunakan hingga 50 anotasi per jejak.

Untuk merekam anotasi di luar fungsi AWS Lambda

1. Dapatkan instans dari `AWSXRayRecorder`.

```
using Amazon.XRay.Recorder.Core;  
...  
AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
```

2. Panggil `addAnnotation` dengan kunci `String` dan `Boolean`, `Int32`, `Int64`, `Double`, atau nilai `String`.

```
recorder.AddAnnotation("mykey", "my value");
```

Contoh berikut menunjukkan cara memanggil `putAnnotation` dengan kunci `String` yang mencakup titik, dan nilai `Boolean`, `Number`, atau `String`.

```
document.putAnnotation("testkey.test", "my value");
```

Untuk merekam anotasi di dalam fungsi AWS Lambda

Segmen dan subsegmen di dalam fungsi Lambda dikelola oleh lingkungan runtime Lambda. Jika Anda ingin menambahkan anotasi ke segmen atau subsegmen di dalam fungsi Lambda, Anda harus melakukan hal berikut:

1. Buat segmen atau subsegmen di dalam fungsi Lambda.
2. Tambahkan anotasi ke segmen atau subsegmen.
3. Akhiri segmen atau subsegmen.

Contoh kode berikut menunjukkan cara menambahkan anotasi ke subsegmen di dalam fungsi 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();
}
```

X-Ray SDK merekam anotasi sebagai pasangan nilai kunci dalam annotations objek dalam dokumen segmen. Memanggil `addAnnotation` operasi dua kali dengan kunci yang sama menimpa nilai yang direkam sebelumnya pada segmen atau subsegmen yang sama.

Untuk menemukan pelacakan yang memiliki anotasi dengan nilai spesifik, gunakan kata kunci `annotation[key]` di [ekspresi filter](#).

Mencatat metadata dengan X-Ray SDK for .NET

Gunakan metadata untuk merekam informasi pada segmen atau subsegmen yang tidak perlu Anda indeks untuk digunakan di dalam pencarian. Nilai metadata dapat berupa string, angka, boolean, atau objek lain yang dapat diserialisasi menjadi objek atau array JSON.

Untuk mencatat metadata

1. Dapatkan instance dari `AWSXRayRecorder`, seperti yang ditunjukkan dalam contoh kode berikut:

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
```

2. Panggil AddMetadata dengan namespace string, kunci string, dan nilai objek, seperti yang ditunjukkan pada contoh kode berikut:

```
recorder.AddMetadata("my namespace", "my key", "my value");
```

Anda juga dapat memanggil AddMetadata operasi hanya menggunakan pasangan kunci dan nilai, seperti yang ditunjukkan pada contoh kode berikut:

```
recorder.AddMetadata("my key", "my value");
```

Jika Anda tidak menentukan nilai untuk namespace, X-Ray SDK akan digunakan. default. Memanggil AddMetadata operasi dua kali dengan kunci yang sama menimpa nilai yang direkam sebelumnya pada segmen atau subsegmen yang sama.

Bekerja dengan Ruby

Ada dua cara untuk instrumen aplikasi Ruby Anda untuk mengirim jejak ke X-Ray:

- [AWS Distro untuk OpenTelemetry Ruby — AWS Distribusi yang menyediakan serangkaian pustaka open source untuk mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan, termasuk Amazon, AWS X-Ray dan Amazon OpenSearch Service CloudWatch, melalui Distro for Collector.AWS OpenTelemetry](#)
- [AWS X-Ray SDK for Ruby - Satu set perpustakaan untuk menghasilkan dan mengirim jejak ke X-Ray melalui daemon X-Ray.](#)

Untuk informasi selengkapnya, lihat [Memilih antara AWS Distro untuk OpenTelemetry dan X-Ray SDKs](#).

AWS Distro para Ruby OpenTelemetry

Dengan AWS Distro for OpenTelemetry (ADOT) Ruby, Anda dapat menginstruksikan aplikasi Anda sekali dan mengirim metrik dan jejak yang berkorelasi ke beberapa AWS solusi pemantauan termasuk Amazon, CloudWatch, AWS X-Ray dan Amazon Service. OpenSearch Menggunakan X-Ray dengan ADOT memerlukan dua komponen: OpenTelemetry SDK diaktifkan untuk digunakan dengan X-Ray, dan AWS Distro untuk OpenTelemetry Kolektor diaktifkan untuk digunakan dengan X-Ray.

Untuk memulai, lihat dokumentasi [AWS Distro untuk OpenTelemetry Ruby](#).

Untuk informasi selengkapnya tentang menggunakan AWS Distro untuk OpenTelemetry with AWS X-Ray dan lainnya Layanan AWS, lihat [AWS Distro untuk OpenTelemetry atau Distro untuk AWS Dokumentasi](#). OpenTelemetry

Untuk informasi selengkapnya tentang dukungan dan penggunaan bahasa, lihat [AWS Observabilitas di GitHub](#).

AWS X-Ray SDK for Ruby

X-Ray SDK for Ruby adalah serangkaian pustaka untuk aplikasi web Java yang menyediakan kelas dan metode untuk menghasilkan dan mengirim penelusuran data ke daemon X-Ray. Data pelacakan

mencakup informasi tentang permintaan HTTP masuk yang disajikan oleh aplikasi, dan panggilan yang dilakukan aplikasi ke layanan hilir menggunakan AWS SDK, klien HTTP, atau klien rekaman aktif. Anda juga dapat membuat segmen secara manual dan menambahkan informasi debug dalam anotasi dan metadata.

Anda dapat mengunduh SDK dengan menambahkannya ke gemfile Anda dan menjalankan `bundle install`.

Example Gemfile

```
gem 'aws-sdk'
```

Jika Anda menggunakan Rails, mulailah dengan [Menambahkan middleware SDK X-Ray](#) untuk menelusuri permintaan yang masuk. Sebuah filter permintaan membuat [segmen](#). Ketika segmen terbuka, Anda dapat menggunakan metode klien SDK untuk menambahkan informasi ke segmen dan membuat subsegmen untuk penelusuran panggilan hilir. SDK juga secara otomatis mencatat pengecualian bahwa aplikasi Anda melempar sementara segmen terbuka. Untuk aplikasi Non-Rail, Anda dapat [membuat segmen secara manual](#).

Selanjutnya, gunakan X-Ray SDK untuk instrumen klien AWS SDK untuk Ruby, HTTP, dan SQL Anda dengan [mengonfigurasi perekam](#) untuk menambal pustaka terkait. Setiap kali Anda melakukan panggilan ke hilir Layanan AWS atau sumber daya dengan klien yang diinstrumentasi, SDK akan mencatat informasi tentang panggilan di subsegmen. Layanan AWS dan sumber daya yang Anda akses dalam layanan muncul sebagai node hilir pada peta jejak untuk membantu Anda mengidentifikasi kesalahan dan masalah pembatasan pada koneksi individual.

Setelah menjalankan SDK, sesuaikan perilakunya dengan [mengonfigurasi pencatat dan penanganan pesan](#). Anda dapat menambahkan plugin untuk mencatat data mengenai sumber daya komputasi yang berjalan di aplikasi Anda, menyesuaikan perilaku sampling dengan mendefinisikan aturan sampling, dan mengatur tingkat log untuk melihat lebih atau kurang informasi dari SDK dalam log aplikasi Anda.

Catat informasi tambahan tentang permintaan dan pekerjaan yang dilakukan aplikasi Anda dalam [anotasi dan metadata](#). Anotasi adalah pasangan kunci-nilai sederhana yang diindeks untuk digunakan dengan [ekspresi filter](#), sehingga Anda dapat mencari pelacakan yang berisi data tertentu. Entri metadata kurang bersifat membatasi dan dapat mencatat seluruh objek dan array — segala yang dapat disambungkan ke dalam JSON.

Anotasi dan Metadata

Anotasi dan metadata adalah teks abstrak yang Anda tambahkan ke segmen dengan X-Ray SDK. Anotasi diindekskan untuk digunakan dengan ekspresi filter. Metadata tidak diindeks, tetapi dapat dilihat di segmen mentah dengan konsol X-Ray atau API. Siapa pun yang Anda berikan akses baca ke X-Ray dapat melihat data ini.

Bila Anda memiliki banyak klien diinstrumentasi dalam kode Anda, segmen permintaan tunggal dapat berisi sejumlah besar subsegmen, satu untuk setiap panggilan yang dilakukan dengan klien yang diinstrumentasi. Anda dapat mengatur dan mengelompokkan subsegmen dengan menggabungkan panggilan klien di [subsegmen kustom](#). Anda dapat membuat subsegmen kustom untuk seluruh fungsi atau bagian dari kode, dan mencatat metadata dan anotasi pada subsegmen bukan menulis segala sesuatu pada segmen induk.

Untuk dokumentasi referensi kelas SDK dan metode, lihat [AWS X-Ray Referensi API SDK for Ruby](#).

Persyaratan

SDK X-Ray memerlukan Ruby 2.3 atau yang lebih baru dan kompatibel dengan pustaka berikut ini:

- AWS SDK untuk Ruby versi 3.0 atau yang lebih baru
- Versi rail 5.1 atau lebih baru

Mengonfigurasi X-Ray SDK for Ruby

X-Ray SDK for Ruby mencakup kelas bernama `XRayRecorder` yang menyediakan pencatat global. Anda dapat mengonfigurasi pencatat global untuk menyesuaikan yang membuat segmen untuk panggilan HTTP masuk.

Bagian-bagian

- [Plugin layanan](#)
- [Aturan pengambilan sampel](#)
- [Pencatatan log](#)
- [Konfigurasi pencatat dalam kode](#)
- [Konfigurasi pencatat dengan rail](#)

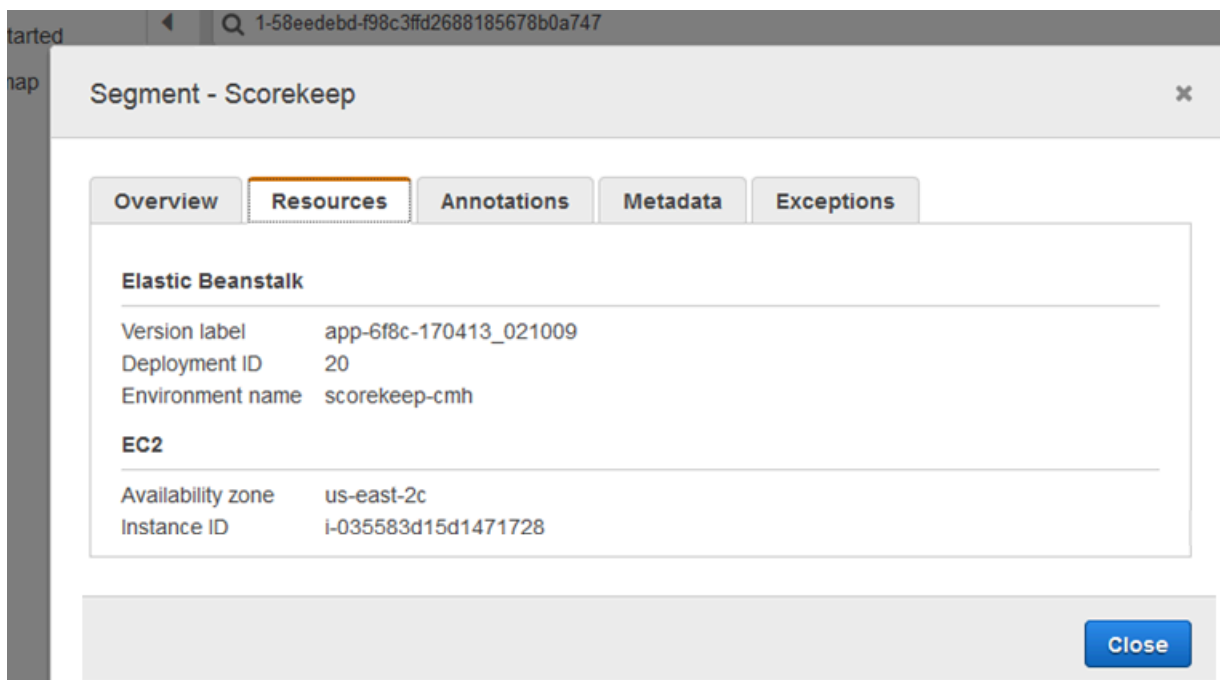
- [Variabel-variabel lingkungan](#)

Plugin layanan

Gunakan plugins untuk mencatat informasi tentang layanan yang meng-hosting aplikasi Anda.

Plugin

- Amazon EC2 — `ec2` menambahkan ID instans dan Availability Zone.
- Elastic Beanstalk – `elastic_beanstalk` menambahkan nama lingkungan, label versi, dan ID deployment.
- Amazon ECS – `ecs` menambahkan ID container.



Untuk menggunakan plugin, tentukan di objek konfigurasi yang Anda lewati ke pencatat.

Example main.rb – Konfigurasi plugin

```
my_plugins = %I[ec2 elastic_beanstalk]

config = {
  plugins: my_plugins,
  name: 'my app',
}
```

```
XRay.recorder.configure(config)
```

Anda juga dapat menggunakan [variabel lingkungan](#), yang diutamakan daripada nilai-nilai yang ditetapkan dalam kode, untuk mengonfigurasi pencatat.

SDK juga menggunakan pengaturan plugin untuk mengatur bidang `origin` pada segmen. Ini menunjukkan jenis AWS sumber daya yang menjalankan aplikasi Anda. Saat Anda menggunakan beberapa plugin, SDK menggunakan urutan resolusi berikut untuk menentukan asal: ElasticBeanstalk > EKS > ECS > EC2

Aturan pengambilan sampel

SDK menggunakan aturan pengambilan sampel yang Anda tetapkan di konsol X-Ray untuk menentukan permintaan yang akan dicatat. Aturan default menelusuri permintaan pertama setiap detik, dan lima persen permintaan tambahan di semua layanan yang mengirim pelacakan ke X-Ray. [Buat aturan tambahan di konsol X-Ray](#) untuk menyesuaikan jumlah data yang dicatat untuk setiap aplikasi Anda.

SDK menerapkan aturan kustom sesuai urutan penetapannya. Jika permintaan cocok dengan beberapa aturan kustom, SDK hanya menerapkan aturan pertama.

Note

Jika SDK tidak dapat mencapai X-Ray untuk mendapatkan aturan pengambilan sampel, SDK akan beralih ke aturan lokal default dari permintaan pertama setiap detik, dan lima persen permintaan tambahan per host. Hal ini dapat terjadi jika host tidak memiliki izin untuk memanggil sampling APIs, atau tidak dapat terhubung ke daemon X-Ray, yang bertindak sebagai proxy TCP untuk panggilan API yang dilakukan oleh SDK.

Anda juga dapat mengonfigurasi SDK untuk memuat aturan sampling dari dokumen JSON. SDK dapat menggunakan aturan lokal sebagai cadangan jika terjadi kasus tidak dapat mengambil sampel X-Ray, atau menggunakan aturan lokal secara eksklusif.

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
}
```

Contoh ini menentukan satu aturan kustom dan aturan default. Aturan kustom menerapkan tingkat pengambilan sampel lima persen tanpa jumlah minimum permintaan untuk melacak jalur di `/api/move/`. Aturan default menelusuri permintaan pertama setiap detik dan 10 persen dari permintaan tambahan.

Kerugian dari menentukan aturan secara lokal adalah bahwa target tetap diterapkan oleh setiap instans pencatat secara independen, alih-alih dikelola oleh layanan X-Ray. Ketika Anda men-deploy lebih banyak host, laju tetap akan dikalikan, sehingga sulit untuk mengontrol jumlah data yang dicatat.

Untuk mengonfigurasi aturan cadangan, menentukan hash untuk dokumen di objek konfigurasi yang Anda berikan ke pencatat.

Example main.rb - Konfigurasi aturan pencadangan

```
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)
```

Untuk menyimpan aturan sampling secara independen, menentukan hash dalam file terpisah dan memerlukan file untuk menariknya ke dalam aplikasi Anda.

Example config/sampling-rules.rb

```
my_sampling_rules = {  
  version: 1,  
  default: {  
    fixed_target: 1,  
    rate: 0.1  
  }  
}
```

Example main.rb - Aturan sampling dari file

```
require 'aws-xray-sdk'  
require 'config/sampling-rules.rb'  
  
config = {  
  sampling_rules: my_sampling_rules,  
  name: 'my app',  
}  
XRay.recorder.configure(config)
```

Untuk menggunakan hanya aturan lokal, memerlukan aturan pengambilan sampel dan mengonfigurasi LocalSampler.

Example main.rb - Aturan pengambilan sampel lokal

```
require 'aws-xray-sdk'  
require 'aws-xray-sdk/sampling/local/sampler'  
  
config = {  
  sampler: LocalSampler.new,  
  name: 'my app',  
}  
XRay.recorder.configure(config)
```

Anda juga dapat mengonfigurasi pencatat global untuk menonaktifkan pengambilan sampel dan instrumen semua permintaan masuk.

Example main.rb – Nonaktifkan pengambilan sampel

```
require 'aws-xray-sdk'
config = {
  sampling: false,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Pencatatan log

Secara default, pencatat output peristiwa tingkat info untuk `$stdout`. Anda dapat menyesuaikan log dengan mendefinisikan [Pencatat](#) di objek konfigurasi yang Anda lulus ke pencatat.

Example main.rb – Logging

```
require 'aws-xray-sdk'
config = {
  logger: my_logger,
  name: 'my app',
}
XRay.recorder.configure(config)
```

Gunakan log debug untuk mengidentifikasi masalah, seperti subsegmen yang tidak tertutup, saat Anda [menghasilkan subsegmen secara manual](#).

Konfigurasi pencatat dalam kode

Pengaturan tambahan tersedia dari metode `configure` pada `XRay.recorder`.

- `context_missing` – Mengatur ke `LOG_ERROR` untuk menghindari mengembalikan pengecualian ketika kode instrumentasi Anda mencoba mencatat data ketika tidak ada segmen yang terbuka.
- `daemon_address` – Mengatur host dan port listener daemon X-Ray.
- `name` – Atur nama layanan yang digunakan SDK untuk segmen.
- `naming_pattern` – Atur pola nama domain untuk menggunakan [Penamaan dinamis](#).
- `plugins` – Catat informasi tentang sumber daya AWS aplikasi dengan [plugin](#).
- `sampling` – Atur ke `false` untuk menonaktifkan pengambilan sampel.
- `sampling_rules` – Atur hash yang berisi [Aturan pengambilan sampel](#).

Example main.rb - Menonaktifkan konteks pengecualian hilang

```
require 'aws-xray-sdk'
config = {
  context_missing: 'LOG_ERROR'
}

XRay.recorder.configure(config)
```

Konfigurasi pencatat dengan rail

Jika Anda menggunakan kerangka kerja Rails, Anda dapat mengonfigurasi opsi pada pencatat global dalam file Ruby di bawah `app_root/initializers`. SDK X-Ray mendukung kunci konfigurasi tambahan untuk digunakan dengan Rails.

- `active_record` – Atur ke `true` untuk mencatat subsegmen untuk transaksi basis data Catatan Aktif.

Mengonfigurasi pengaturan yang tersedia di objek konfigurasi bernama `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
}
```

Variabel-variabel lingkungan

Anda dapat menggunakan variabel lingkungan untuk mengonfigurasi X-Ray SDK for Ruby. SDK mendukung variabel berikut:

- `AWS_XRAY_TRACING_NAME` – Atur nama layanan yang digunakan SDK untuk segmen. Menimpa nama layanan yang Anda tetapkan pada [strategi penamaan segmen](#) filter servlet.
- `AWS_XRAY_DAEMON_ADDRESS` – Atur host dan port pendengar daemon X-Ray. Secara default, SDK mengirimkan pelacakan data ke `127.0.0.1:2000`. Gunakan variabel ini jika Anda telah mengonfigurasi daemon untuk [mendengarkan pada port yang berbeda](#) atau jika berjalan pada host yang berbeda.

- `AWS_XRAY_CONTEXT_MISSING`— Setel `RUNTIME_ERROR` untuk melempar pengecualian saat kode instrumentasi Anda mencoba merekam data saat tidak ada segmen yang terbuka.

Nilai Valid

- `RUNTIME_ERROR`— Lempar pengecualian runtime.
- `LOG_ERROR`— Log kesalahan dan lanjutkan (default).
- `IGNORE_ERROR`— Abaikan kesalahan dan lanjutkan.

Kesalahan yang berkaitan dengan segmen atau subsegmen yang hilang dapat terjadi ketika Anda mencoba untuk menggunakan klien yang diinstrumentasi dalam kode perusahaan rintisan yang berjalan ketika tidak ada permintaan terbuka, atau dalam kode yang memunculkan thread baru.

variabel lingkungan menimpa nilai yang ditetapkan dalam kode.

Menelusuri permintaan yang masuk dengan X-Ray SDK for Ruby

Anda dapat menggunakan X-Ray SDK untuk melacak permintaan HTTP masuk yang dilayani aplikasi Anda pada EC2 instans di Amazon EC2 AWS Elastic Beanstalk, atau Amazon ECS.

Jika Anda menggunakan Rails, gunakan middleware Rails untuk instrumen permintaan HTTP masuk. Ketika Anda menambahkan perangkat tengah X-Ray ke aplikasi Anda, X-Ray SDK for Ruby membuat segmen untuk setiap permintaan sampel. Setiap segmen yang dibuat oleh instrumentasi tambahan menjadi subsegment dari segmen tingkat permintaan yang memberikan informasi tentang permintaan HTTP dan respons. Informasi ini mencakup waktu, metode, dan disposisi permintaan.

Setiap segmen memiliki nama yang mengidentifikasi aplikasi Anda dalam peta layanan. Segmen dapat diberi nama secara statis, atau Anda dapat mengonfigurasi SDK untuk nama itu secara dinamis berdasarkan header host dalam permintaan masuk. Penamaan dinamis memungkinkan Anda mengelompokkan pelacakan berdasarkan nama domain dalam permintaan, dan menerapkan nama default jika nama tersebut tidak cocok dengan pola yang diharapkan (misalnya, jika header host ditiru).

Permintaan yang Diteruskan

Jika penyeimbang beban atau perantara lainnya meneruskan permintaan ke aplikasi Anda, X-Ray akan mengambil IP klien dari header `X-Forwarded-For` dalam permintaan bukan

dari sumber IP dalam paket IP. IP klien yang dicatat untuk permintaan yang diteruskan dapat ditiru, sehingga tidak dapat dipercaya.

Ketika permintaan diteruskan, SDK menetapkan bidang tambahan di segmen untuk menunjukkan ini. Jika segmen berisi bidang `x_forwarded_for` diatur ke `true`, IP klien diambil dari header `X-Forwarded-For` dalam permintaan HTTP.

Perangkat tengah membuat segmen untuk setiap permintaan masuk dengan blok `http` yang berisi informasi berikut:

- Metode HTTP – GET, POST, PUT, DELETE, dll
- Alamat klien – Alamat IP klien yang mengirim permintaan.
- Kode respons – Kode respons HTTP untuk permintaan yang selesai.
- Timing – Waktu mulai (saat permintaan diterima) dan waktu akhir (saat respons dikirim).
- Agen pengguna — `user-agent` dari permintaan.
- Panjang konten — `content-length` dari respons.

Menggunakan middleware rail

Untuk menggunakan middleware, perbarui `gemfile` Anda untuk menyertakan [railtie](#).

Example Gemfile - rail

```
gem 'aws-xray-sdk', require: ['aws-xray-sdk/facets/rails/railtie']
```

Untuk menggunakan middleware, Anda juga harus [mengonfigurasi perekam](#) dengan nama yang mewakili aplikasi di peta jejak.

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {  
  name: 'my app'  
}
```


Mengonfigurasi kode secara manual

Jika Anda tidak menggunakan Rails, buat segmen secara manual. Anda dapat membuat segmen untuk setiap permintaan yang masuk, atau membuat segmen di sekitar klien HTTP atau AWS SDK yang ditambah untuk menyediakan konteks bagi perekam untuk menambahkan subsegmen.

```
# 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
```

Mengonfigurasi strategi penamaan segmen

AWS X-Ray menggunakan nama layanan untuk mengidentifikasi aplikasi Anda dan membedakannya dari aplikasi lain, database, eksternal APIs, dan AWS sumber daya yang digunakan aplikasi Anda. Saat SDK X-Ray membuat segmen untuk permintaan masuk, SDK akan mencatat nama layanan aplikasi Anda di [kolom nama](#).

SDK X-Ray dapat memberi nama segmen setelah nama host di header permintaan HTTP. Namun, header ini dapat ditiru, yang dapat mengakibatkan simpul tak terduga di peta layanan Anda. Untuk mencegah SDK dari penamaan segmen salah karena permintaan dengan header host palsu, Anda harus menentukan nama default untuk permintaan masuk.

Jika aplikasi Anda menyuguhkan permintaan untuk beberapa domain, Anda dapat mengonfigurasi SDK untuk menggunakan strategi penamaan dinamis untuk mencerminkan ini dalam nama segmen.

Strategi penamaan dinamis mengizinkan SDK menggunakan nama host untuk permintaan yang sesuai dengan pola yang diharapkan, dan menerapkan nama default untuk permintaan yang tidak sesuai.

Misalnya, Anda boleh memiliki satu aplikasi yang melayani permintaan untuk tiga subdomain—`www.example.com`, `api.example.com`, dan `static.example.com`. Anda dapat menggunakan strategi penamaan dinamis dengan pola `*.example.com` untuk mengidentifikasi segmen untuk setiap subdomain dengan nama yang berbeda, mengakibatkan tiga simpul layanan pada peta layanan. Jika aplikasi Anda menerima permintaan dengan nama host yang tidak cocok dengan pola, Anda akan melihat simpul keempat pada peta layanan dengan nama fallback yang Anda tentukan.

Untuk menggunakan nama yang sama untuk semua segmen permintaan, tentukan nama aplikasi Anda ketika mengonfigurasi pencatat, seperti yang ditunjukkan di [bagian sebelumnya](#).

Strategi penamaan dinamis menentukan pola yang harus sesuai dengan nama host, dan nama default untuk digunakan jika nama host dalam permintaan HTTP tidak cocok dengan pola. Untuk nama segmen dinamis, menentukan pola penamaan di hash config.

Example main.rb – Penamaan dinamis

```
config = {  
  naming_pattern: '*mydomain*',  
  name: 'my app',  
}  
  
XRay.recorder.configure(config)
```

Anda dapat menggunakan `*` dalam pola untuk mencocokkan string apa pun, atau `?` untuk mencocokkan setiap karakter tunggal.

Note

Anda dapat menyimpan nama layanan default yang Anda tentukan dalam kode dengan [variabel lingkungan](#) `AWS_XRAY_TRACING_NAME`

Pustaka patching ke instrumen panggilan hilir

Untuk instrumen panggilan hilir, gunakan X-Ray SDK for Ruby untuk patch pustaka yang menggunakan aplikasi Anda. X-Ray SDK for Ruby dapat patch pustaka berikut.

Pustaka Didukung

- [net/http](#) – Instrumen klien HTTP.
- [aws-sdk](#)— AWS SDK untuk Ruby Klien instrumen.

Ketika Anda menggunakan pustaka yang dipatch, X-Ray SDK for Ruby membuat subsegmen untuk panggilan dan catatan informasi dari permintaan dan respons. Segmen harus tersedia untuk SDK untuk membuat subsegmen, baik dari SDK middleware atau panggilan ke `XRay.recorder.begin_segment`.

Untuk patch pustaka, tentukan di objek konfigurasi yang Anda lewati ke pencatat X-Ray.

Example main.rb – Patch pustaka

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http aws_sdk]
}

XRay.recorder.configure(config)
```

Menelusuri panggilan AWS SDK dengan X-Ray SDK for Ruby

[Saat aplikasi Anda melakukan panggilan Layanan AWS untuk menyimpan data, menulis ke antrian, atau mengirim notifikasi, X-Ray SDK for Ruby melacak panggilan hilir di subsegmen.](#) Ditelusuri Layanan AWS dan sumber daya yang Anda akses dalam layanan tersebut (misalnya, bucket Amazon S3 atau antrean Amazon SQS), muncul sebagai node hilir pada peta jejak di konsol X-Ray.

X-Ray SDK for Ruby secara otomatis AWS menginstrumentasikan semua klien SDK [saat](#) Anda menambal pustaka. `aws-sdk` Anda tidak dapat instrumen klien individu.

Untuk semua layanan, Anda dapat melihat nama panggilan API di konsol X-Ray. Untuk subset layanan, X-Ray SDK menambahkan informasi ke segmen untuk memberikan lebih banyak perincian di peta layanan.

Sebagai contoh, ketika Anda melakukan panggilan dengan klien DynamoDB berinstrumen, SDK menambahkan nama tabel ke segmen untuk panggilan yang menargetkan tabel. Di konsol tersebut, setiap tabel muncul sebagai simpul terpisah di peta layanan, dengan simpul DynamoDB generik untuk panggilan yang tidak menargetkan tabel.

Example Subsegmen untuk panggilan ke DynamoDB untuk menyimpan item

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNS05AEM8T4FDA4RQDEB940VTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

Ketika Anda mengakses sumber daya bernama, panggilan ke layanan berikut membuat simpul tambahan di peta layanan. Panggilan yang tidak menargetkan sumber daya tertentu membuat simpul generik untuk layanan tersebut.

- Amazon DynamoDB – Nama tabel
- Amazon Simple Storage Service – Bucket dan nama kunci
- Amazon Simple Queue Service – Nama antrean

Menghasilkan subsegmen kustom dengan X-Ray SDK

Subsegmen memperpanjang [segmen](#) penelusuran dengan detail tentang pekerjaan yang dilakukan untuk melayani permintaan. Setiap kali Anda melakukan panggilan dengan klien berinstrumen, X-Ray tersebut mencatat informasi yang dihasilkan dalam subsegmen. Anda dapat membuat subsegment tambahan untuk mengelompokkan subsegment lain, untuk mengukur performa bagian kode, atau untuk mencatat anotasi dan metadata.

Untuk mengelola subsegmen, gunakan metode `begin_subsegment` dan `end_subsegment`

```
subsegment = XRay.recorder.begin_subsegment name: 'annotations', namespace: 'remote'
my_annotations = { id: 12345 }
subsegment.annotations.update my_annotations
XRay.recorder.end_subsegment
```

Untuk membuat subsegmen karena fungsi, membungkusnya dalam panggilan ke `XRay.recorder.capture`.

```
XRay.recorder.capture('name_for_subsegment') do |subsegment|
  resp = myfunc() # myfunc is your function
  subsegment.annotations.update k1: 'v1'
  resp
end
```

Ketika Anda membuat subsegmen dalam segmen atau subsegmen lain, X-Ray SDK menghasilkan ID untuknya dan mencatat waktu mulai dan waktu akhir.

Example Subsegmen dengan metadata

```
"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"
    }
  },
},
```

Tambahkan anotasi dan metadata ke segmen dengan X-Ray SDK for Ruby

Anda dapat mencatat informasi tambahan tentang permintaan, lingkungan, atau aplikasi dengan anotasi dan metadata. Anda dapat menambahkan anotasi dan metadata ke segmen yang dibuat oleh SDK X-Ray, atau subsegmen kustom yang Anda buat.

Anotasi adalah pasangan kunci-nilai dengan string, nomor, atau nilai-nilai Boolean. Anotasi diindekskan untuk digunakan dengan [Ekspresi filter](#). Gunakan anotasi untuk mencatat data yang ingin Anda gunakan untuk mengelompokkan pelacakan di konsol tersebut, atau saat memanggil API [GetTraceSummaries](#).

Metadata adalah pasangan kunci-nilai yang dapat memiliki nilai dari setiap tipe, termasuk objek dan daftar, tetapi tidak diindekskan untuk digunakan dengan ekspresi filter. Gunakan metadata untuk mencatat data tambahan yang ingin disimpan dalam pelacakan tetapi tidak perlu digunakan dengan pencarian.

Selain anotasi dan metadata, Anda juga dapat [mencatat string ID pengguna](#) pada segmen. Pengguna IDs direkam dalam bidang terpisah pada segmen dan diindeks untuk digunakan dengan pencarian.

Bagian-bagian

- [Mencatat pencatatan dengan X-Ray SDK for Ruby](#)
- [Mencatat metadata dengan X-Ray SDK for Ruby](#)
- [Merekam pengguna IDs dengan X-Ray SDK for Ruby](#)

Mencatat pencatatan dengan X-Ray SDK for Ruby

Gunakan anotasi untuk mencatat informasi pada segmen atau subsegmen yang ingin diindeks untuk pencarian.

Persyaratan Anotasi

- Tombol — Kunci untuk anotasi X-Ray dapat memiliki hingga 500 karakter alfanumerik. Anda tidak dapat menggunakan spasi atau simbol selain titik atau titik (.).
- Nilai — Nilai untuk anotasi X-Ray dapat memiliki hingga 1.000 karakter Unicode.
- Jumlah Anotasi — Anda dapat menggunakan hingga 50 anotasi per jejak.

Untuk mencatat anotasi

1. Dapatkan referensi ke segmen atau subsegmen saat ini dari `xray_recorder`.

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

atau

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_subsegment
```

2. Panggilan update dengan nilai hash.

```
my_annotations = { id: 12345 }
document.annotations.update my_annotations
```

Berikut ini adalah contoh yang menunjukkan cara memanggil update dengan kunci anotasi yang berisi titik.

```
my_annotations = { testkey.test: 12345 }
document.annotations.update my_annotations
```

SDK mencatat penjelasan sebagai pasangan nilai kunci dalam objek `annotations` di dokumen segmen. Memanggil `add_annotations` dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Untuk menemukan penelusuran yang memiliki anotasi dengan nilai-nilai tertentu, gunakan kata kunci `annotation[key]` dalam [ekspresi filter](#).

Mencatat metadata dengan X-Ray SDK for Ruby

Gunakan metadata untuk mencatat informasi pada segmen atau subsegmen yang tidak perlu diindeks untuk pencarian. Nilai metadata bisa string, angka, Boolean, atau objek yang dapat diserialkan ke dalam objek JSON atau array.

Untuk mencatat metadata

1. Dapatkan referensi ke segmen atau subsegmen saat ini dari `xray_recorder`.

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

atau

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_subsegment
```

2. Panggil metadata dengan kunci String, Boolean, Nomor, String, atau nilai Objek; dan namespace String.

```
my_metadata = {
  my_namespace: {
    key: 'value'
  }
}
subsegment.metadata my_metadata
```

Memanggil metadata dua kali dengan tombol yang sama akan menimpa nilai yang tercatat sebelumnya pada segmen atau subsegmen yang sama.

Merekam pengguna IDs dengan X-Ray SDK for Ruby

Rekam segmen pengguna IDs berdasarkan permintaan untuk mengidentifikasi pengguna yang mengirim permintaan.

Untuk merekam pengguna IDs

1. Dapatkan referensi ke segmen atau subsegmen saat ini dari `xray_recorder`.

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```


2. Mengatur bidang pengguna pada segmen untuk ID String dari pengguna yang mengirim permintaan.

```
segment.user = 'U12345'
```

Anda dapat menelepon di pengendali Anda untuk mencatat ID pengguna segera setelah aplikasi mulai memproses permintaan.

Untuk menemukan penelusuran pada ID pengguna, gunakan kata kunci `user` dalam [ekspresi filter](#).

Migrasi dari instrumentasi X-Ray ke instrumentasi OpenTelemetry

X-Ray beralih ke OpenTelemetry (OTel) sebagai standar instrumentasi utamanya untuk penelusuran aplikasi dan observabilitas. Pergeseran strategis ini sejalan AWS dengan praktik terbaik industri dan menawarkan pelanggan solusi yang lebih komprehensif, fleksibel, dan siap di masa depan untuk kebutuhan pengamatan mereka. OpenTelemetryAdopsi luas di industri memungkinkan penelusuran permintaan di berbagai sistem, termasuk yang di luar AWS yang mungkin tidak secara langsung terintegrasi dengan X-Ray.

Bab ini memberikan rekomendasi untuk transisi yang mulus, dan menekankan pentingnya migrasi ke solusi OpenTelemetry berbasis untuk memastikan dukungan berkelanjutan dan akses ke fitur-fitur terbaru dalam instrumentasi dan observabilitas aplikasi.

Disarankan untuk mengadopsi OpenTelemetry sebagai solusi observabilitas untuk menginstrumentasi aplikasi Anda.

Topik

- [Memahami OpenTelemetry](#)
- [Memahami OpenTelemetry konsep migrasi](#)
- [Ikhtisar migrasi](#)
- [Migrasi dari X-Ray Daemon ke AWS CloudWatch agen atau kolektor OpenTelemetry](#)
- [Migrasi ke OpenTelemetry Java](#)
- [Migrasi ke Go OpenTelemetry](#)
- [Migrasi ke Node.js OpenTelemetry](#)
- [Migrasi ke OpenTelemetry .NET](#)
- [Migrasi ke Python OpenTelemetry](#)
- [Migrasi ke Ruby OpenTelemetry](#)

Memahami OpenTelemetry

OpenTelemetry adalah kerangka observabilitas standar industri yang menyediakan protokol dan alat standar untuk mengumpulkan data telemetri. Ini menawarkan pendekatan terpadu untuk

menginstrumentasi, menghasilkan, mengumpulkan, dan mengekspor data telemetri seperti metrik, log, dan jejak.

Ketika Anda bermigrasi dari X-Ray SDKs ke OpenTelemetry, Anda mendapatkan manfaat berikut:

- Kerangka kerja yang ditingkatkan dan dukungan instrumentasi perpustakaan
- Support untuk bahasa pemrograman tambahan
- Kemampuan instrumentasi otomatis
- Opsi konfigurasi pengambilan sampel yang fleksibel
- Koleksi metrik, log, dan jejak terpadu

OpenTelemetry Kolektor menyediakan lebih banyak opsi untuk format pengumpulan data dan tujuan ekspor daripada daemon X-Ray.

OpenTelemetry dukungan di AWS

AWS menyediakan beberapa solusi untuk bekerja dengan OpenTelemetry:

- AWS Distro untuk OpenTelemetry

Ekspor OpenTelemetry jejak sebagai segmen ke X-Ray.

Untuk informasi lebih lanjut, lihat [AWS Distro untuk OpenTelemetry](#).

- CloudWatch Sinyal Aplikasi

Ekspor OpenTelemetry jejak dan metrik yang disesuaikan untuk memantau kesehatan aplikasi.

Untuk informasi selengkapnya, lihat [Bekerja dengan Sinyal Aplikasi](#).

- CloudWatch OTel Titik akhir

Ekspor OpenTelemetry jejak ke X-Ray menggunakan OTel titik akhir HTTP dengan OpenTelemetry instrumentasi asli.

Untuk informasi selengkapnya, lihat [Menggunakan OTel titik akhir](#).

Menggunakan OpenTelemetry dengan AWS CloudWatch

AWS CloudWatch mendukung OpenTelemetry jejak melalui instrumentasi aplikasi sisi klien dan AWS CloudWatch layanan asli seperti Sinyal Aplikasi, Jejak, Peta, Metrik, dan Log. Untuk informasi selengkapnya, lihat [OpenTelemetry](#).

Memahami OpenTelemetry konsep migrasi

Tabel berikut memetakan konsep X-Ray ke OpenTelemetry padanannya. Memahami pemetaan ini membantu Anda menerjemahkan instrumentasi X-Ray yang ada ke: OpenTelemetry

Konsep X-Ray	OpenTelemetry konsep
Perekam X-Ray	Penyedia dan Pelacak Pelacak
Plugin Layanan	Detektor Sumber Daya
Segment	Rentang (Server)
Sub-segmen	(Non-server) Rentang
Aturan Pengambilan Sampel X-Ray	OpenTelemetry Sampling (Dapat Disesuaikan)
Pemancar X-Ray	Eksportir Span (Dapat Disesuaikan)
Anotasi/Metadata	Atribut
Instrumentasi Perpustakaan	Instrumentasi Perpustakaan
Konteks Jejak X-Ray	Konteks Rentang
Propagasi Konteks Jejak Sinar-X	Propagasi Konteks Jejak W3C
Pengambilan Sampel Jejak X-Ray	OpenTelemetry Pengambilan Sampel Jejak
N/A	Pemrosesan Rentang
N/A	Bagasi
Daemon X-Ray	OpenTelemetry Kolektor

Note

Untuk informasi lebih lanjut tentang OpenTelemetry konsep, lihat [OpenTelemetry dokumentasi](#).

Membandingkan fitur

Tabel berikut menunjukkan fitur mana yang didukung di kedua layanan. Gunakan informasi ini untuk mengidentifikasi celah yang perlu Anda atasi selama migrasi:

Fitur	Instrumentasi X-Ray	OpenTelemetry instrumentasi
Instrumentasi perpustakaan	Didukung	Didukung
Pengambilan sampel X-Ray	Didukung	Didukung di OTel Java/.net/Go Didukung di ADOT Java/.NET/Python/Node.js
Propagasi konteks jejak X-Ray	Didukung	Didukung
Deteksi sumber daya	Didukung	Didukung
Anotasi segmen	Didukung	Didukung
Metadata segmen	Didukung	Didukung
Instrumentasi otomatis kode nol	Didukung di Jawa	Didukung di OTel Java/. NET/Python/Node.js Didukung di ADOT Java/.NET/Python/Node.js
Lacak pembuatan secara manual	Didukung	Didukung

Menyiapkan dan mengonfigurasi penelusuran

Untuk membuat jejak OpenTelemetry, Anda memerlukan pelacak. Anda mendapatkan pelacak dengan menginisialisasi Penyedia Pelacak di aplikasi Anda. Ini mirip dengan bagaimana Anda menggunakan X-Ray Recorder untuk mengkonfigurasi X-Ray dan membuat segmen dan subsegmen dalam jejak X-Ray.

Note

Penyedia OpenTelemetry Tracer menawarkan lebih banyak opsi konfigurasi daripada Perekam X-Ray.

Memahami struktur data jejak

Setelah memahami konsep dasar dan pemetaan fitur, Anda dapat mempelajari detail implementasi spesifik seperti struktur data jejak dan pengambilan sampel.

OpenTelemetry menggunakan bentang alih-alih segmen dan subsegmen untuk menyusun data jejak. Setiap rentang mencakup komponen-komponen berikut:

- Nama
- ID Unik
- Mulai dan akhiri stempel waktu
- Jenis rentang
- Konteks rentang
- Atribut (metadata nilai kunci)
- Acara (log stempel waktu)
- Tautan ke bentang lainnya
- Informasi status
- Referensi rentang induk

Saat Anda bermigrasi ke OpenTelemetry, rentang Anda secara otomatis dikonversi ke segmen atau subsegmen X-Ray. Ini memastikan pengalaman CloudWatch konsol Anda yang ada tetap tidak berubah.

Bekerja dengan atribut span

X-Ray SDK menyediakan dua cara untuk menambahkan data ke segmen dan subsegmen:

Anotasi

Pasangan nilai kunci yang diindeks untuk pemfilteran dan pencarian

Metadata

Pasangan nilai kunci yang berisi data kompleks yang tidak diindeks untuk pencarian

Secara default, atribut OpenTelemetry rentang dikonversi ke metadata dalam data mentah X-Ray. Untuk mengonversi atribut tertentu menjadi anotasi, tambahkan kuncinya ke daftar `aws.xray.annotations` atribut.

- Untuk informasi lebih lanjut tentang OpenTelemetry konsep, lihat [OpenTelemetry Jejak](#)
- Untuk detail tentang cara memetakan OpenTelemetry data ke data X-Ray, lihat [OpenTelemetry terjemahan model data X-Ray](#)

Mendeteksi sumber daya di lingkungan Anda

OpenTelemetry menggunakan Detektor Sumber Daya untuk mengumpulkan metadata tentang sumber daya yang menghasilkan data telemetri. Metadata ini disimpan sebagai Atribut Sumber Daya. Misalnya, entitas yang memproduksi telemetri dapat berupa kluster Amazon ECS atau EC2 instans Amazon, dan Atribut Sumber Daya yang dapat direkam dari entitas ini dapat menyertakan ARN Cluster Amazon ECS atau ID Instans Amazon. EC2

- Untuk informasi tentang jenis sumber daya yang didukung, lihat Konvensi [Semantik OpenTelemetry Sumber Daya](#)
- Untuk informasi tentang plugin layanan X-Ray, lihat [Mengonfigurasi X-Ray](#) SDK

Mengelola strategi pengambilan sampel

Trace sampling membantu Anda mengelola biaya dengan mengumpulkan data dari subset permintaan yang representatif, bukan semua permintaan. Keduanya OpenTelemetry dan X-Ray mendukung pengambilan sampel, tetapi menerapkannya secara berbeda.

Note

Pengambilan sampel kurang dari 100% jejak mengurangi biaya observabilitas Anda sambil mempertahankan wawasan yang berarti tentang kinerja aplikasi Anda.

OpenTelemetry menyediakan beberapa strategi pengambilan sampel bawaan dan memungkinkan Anda membuat yang khusus. Anda juga dapat mengonfigurasi X-Ray Remote Sampler dalam beberapa bahasa SDK untuk menggunakan aturan sampling X-Ray. OpenTelemetry

Strategi pengambilan sampel tambahan dari OpenTelemetry adalah:

- Pengambilan Sampel Berbasis Orang Tua — Menghormati keputusan pengambilan sampel rentang induk sebelum menerapkan strategi pengambilan sampel tambahan
- Trace ID Ratio Based Sampling -> Sampel secara acak persentase bentang tertentu
- Pengambilan sampel ekor - Menerapkan aturan pengambilan sampel untuk menyelesaikan jejak di Kolektor OpenTelemetry
- Custom sampler - Menerapkan logika sampling Anda sendiri menggunakan antarmuka sampling

Untuk informasi tentang aturan pengambilan sampel X-Ray, lihat [Aturan pengambilan sampel di konsol X-Ray](#)

Untuk informasi tentang pengambilan sampel OpenTelemetry ekor, lihat [Prosesor pengambilan sampel ekor](#)

Mengelola konteks jejak

X-Ray SDKs mengelola Konteks Segmen untuk menangani hubungan induk-anak dengan benar antara Segmen dan Subsegmen dalam jejak. OpenTelemetry menggunakan mekanisme serupa untuk memastikan bahwa bentang memiliki rentang induk yang benar. Ini menyimpan dan menyebarkan data penelusuran di seluruh konteks permintaan. Misalnya, ketika aplikasi Anda memproses permintaan dan membuat rentang server untuk mewakili permintaan itu, OpenTelemetry akan menyimpan rentang server dalam OpenTelemetry Konteks sehingga ketika rentang anak dibuat, rentang anak tersebut dapat mereferensikan rentang dalam Konteks sebagai induknya.

Menyebarkan konteks jejak

Baik X-Ray dan OpenTelemetry menggunakan header HTTP untuk menyebarkan konteks jejak di seluruh layanan. Ini memungkinkan Anda untuk menautkan data jejak yang dihasilkan oleh berbagai layanan dan mempertahankan keputusan pengambilan sampel.

X-Ray SDK secara otomatis menyebarkan konteks jejak menggunakan header jejak X-Ray. Ketika satu layanan memanggil yang lain, header trace berisi konteks yang diperlukan untuk mempertahankan hubungan induk-anak antar jejak.

OpenTelemetry mendukung beberapa format header jejak untuk propagasi konteks, termasuk:

- Konteks Jejak W3C (default)
- Header jejak X-Ray
- Format khusus lainnya

Note

Anda dapat mengonfigurasi OpenTelemetry untuk menggunakan satu atau beberapa format header. Misalnya, gunakan Propagator X-Ray untuk mengirim konteks jejak ke AWS layanan yang mendukung penelusuran X-Ray.

Konfigurasi dan gunakan Propagator X-Ray untuk memungkinkan penelusuran di seluruh AWS layanan. Ini memungkinkan Anda untuk menyebarkan konteks jejak ke titik akhir API Gateway dan layanan lain yang mendukung X-Ray.

- Untuk informasi tentang header jejak X-Ray, lihat [Menelusuri header](#) di Panduan Pengembang X-Ray
- Untuk informasi tentang propagasi OpenTelemetry konteks, lihat [Propagasi Konteks dan Konteks](#) dalam dokumentasi OpenTelemetry

Menggunakan instrumentasi perpustakaan

X-Ray dan OpenTelemetry menyediakan instrumentasi perpustakaan yang memerlukan perubahan kode minimal untuk menambahkan penelusuran ke aplikasi Anda.

X-Ray menyediakan fungsi instrumentasi perpustakaan. Ini memungkinkan Anda untuk menambahkan instrumentasi X-Ray pra-bangun dengan perubahan kode aplikasi minimal. Instrumentasi ini mendukung pustaka tertentu seperti Klien AWS SDK dan HTTP, serta kerangka kerja web seperti Spring Boot atau Express.js.

OpenTelemetryPustaka instrumentasi menghasilkan rentang detail untuk pustaka Anda melalui kait pustaka atau modifikasi kode otomatis, yang memerlukan sedikit perubahan kode.

[Untuk menentukan apakah Library OpenTelemetry Instrumentations mendukung library Anda, cari di OpenTelemetry OpenTelemetry Registry at Registry.](#)

Mengekspor jejak

X-Ray dan OpenTelemetry menggunakan metode yang berbeda untuk mengekspor data jejak.

Ekspor jejak X-Ray

X-Ray SDKs menggunakan emitor untuk mengirim data jejak:

- Mengirim segmen dan subsegmen ke Daemon X-Ray
- Menggunakan UDP untuk I/O non-pemblokiran
- Dikonfigurasi secara default di SDK

OpenTelemetry jejak ekspor

OpenTelemetry menggunakan Eksportir Span yang dapat dikonfigurasi untuk mengirim data jejak:

- Menggunakan protokol http/protobuf atau grpc
- Ekspor mencakup titik akhir yang dipantau oleh Kolektor atau Agen OpenTelemetry CloudWatch
- Memungkinkan konfigurasi eksportir kustom

Memproses dan meneruskan jejak

Baik X-Ray dan OpenTelemetry menyediakan komponen untuk menerima, memproses, dan meneruskan data jejak.

Pemrosesan jejak X-Ray

X-Ray Daemon menangani pemrosesan jejak:

- Mendengarkan lalu lintas UDP dari X-Ray SDKs
- Batch segmen dan subsegmen
- Mengunggah batch ke layanan X-Ray

OpenTelemetry pemrosesan jejak

OpenTelemetry Kolektor menangani pemrosesan jejak:

- Menerima jejak dari layanan berinstrumen
- Memproses dan secara opsional memodifikasi data jejak
- Mengirim jejak yang diproses ke berbagai backend, termasuk X-Ray

Note

AWS CloudWatch Agen juga dapat menerima dan mengirim OpenTelemetry jejak ke X-Ray. Untuk informasi selengkapnya, lihat [Mengumpulkan metrik dan jejak dengan OpenTelemetry](#).

Pemrosesan rentang (OpenTelemetry-konsep spesifik)

OpenTelemetry menggunakan Prosesor Span untuk memodifikasi rentang saat dibuat:

- Memungkinkan membaca dan memodifikasi rentang saat pembuatan atau penyelesaian
- Mengaktifkan logika kustom untuk penanganan rentang

Bagasi (konsep OpenTelemetry -specific)

OpenTelemetryFitur Bagasi memungkinkan penyebaran data nilai kunci:

- Memungkinkan meneruskan data arbitrer di samping konteks pelacakan
- Berguna untuk menyebarkan informasi spesifik aplikasi melintasi batas layanan

Untuk informasi tentang OpenTelemetry Kolektor, lihat [OpenTelemetry Kolektor](#)

Untuk informasi tentang konsep X-Ray, lihat [konsep X-Ray](#) di Panduan Pengembang X-Ray

Ikhtisar migrasi

Bagian ini memberikan gambaran umum tentang perubahan kode yang diperlukan untuk migrasi. Daftar di bawah ini adalah panduan khusus bahasa dan langkah-langkah migrasi Daemon X-Ray.

Important

Untuk sepenuhnya bermigrasi dari instrumentasi X-Ray ke OpenTelemetry instrumentasi, Anda perlu:

1. Ganti penggunaan X-Ray SDK dengan solusi OpenTelemetry
2. Ganti Daemon X-Ray dengan CloudWatch Agen atau OpenTelemetry Kolektor (dengan Eksportir X-Ray)

- [Migrasi ke OpenTelemetry Java](#)
- [Migrasi ke Go OpenTelemetry](#)
- [Migrasi ke Node.js OpenTelemetry](#)
- [Migrasi ke OpenTelemetry .NET](#)
- [Migrasi ke Python OpenTelemetry](#)
- [Migrasi ke Ruby OpenTelemetry](#)

Rekomendasi untuk aplikasi baru dan yang sudah ada

Untuk aplikasi baru dan yang sudah ada, disarankan untuk menggunakan solusi berikut untuk mengaktifkan penelusuran di aplikasi Anda:

Instrumentasi

- OpenTelemetry SDKs
- AWS Distro untuk Instrumentasi OpenTelemetry

Pengumpulan Data

- OpenTelemetry Kolektor
- CloudWatch Agen

Setelah bermigrasi ke solusi OpenTelemetry berbasis, CloudWatch pengalaman Anda akan tetap sama. Anda masih dapat melihat jejak Anda dalam format yang sama di halaman Jejak dan Jejak Peta CloudWatch konsol, atau mengambil data jejak Anda melalui [APIsX-Ray](#).

Menelusuri perubahan pengaturan

Anda perlu mengganti pengaturan X-Ray dengan OpenTelemetry pengaturan.

Perbandingan X-Ray dan OpenTelemetry Pengaturan

Fitur	SDK X-Ray	OpenTelemetry
Konfigurasi default	• Pengambilan Sampel Terpusat X-Ray • Perambatan Konteks Jejak Sinar-X • Lacak Ekspor ke Daemon X-Ray	• Mengekspor jejak ke OpenTelemetry Kolektor atau CloudWatch Agen (HTTP/gRPC) • Propagasi Konteks Jejak W3C
Konfigurasi manual	• Aturan pengambilan sampel lokal • Plug-in deteksi sumber daya	• X-Ray Sampling (mungkin tidak tersedia untuk semua bahasa) • Deteksi sumber daya • Perambatan Konteks Jejak Sinar-X

Perubahan instrumentasi perpustakaan

Perbarui kode Anda untuk menggunakan Instrumentasi OpenTelemetry Perpustakaan alih-alih Instrumentasi Perpustakaan X-Ray untuk AWS SDK, Klien HTTP, Kerangka Web, dan pustaka lainnya. Ini menghasilkan OpenTelemetry Jejak alih-alih Jejak X-Ray.

Note

Perubahan kode bervariasi menurut bahasa dan perpustakaan. Lihat panduan migrasi khusus bahasa untuk petunjuk terperinci.

Perubahan instrumentasi lingkungan Lambda

Untuk digunakan OpenTelemetry dalam fungsi Lambda Anda, pilih salah satu opsi penyiapan ini:

1. Gunakan Layer Lambda instrumentasi otomatis:

- (Disarankan) [Sinyal CloudWatch Aplikasi Lapisan Lambda](#)

Note

Untuk hanya menggunakan penelusuran, atur variabel lingkungan Lambda.

`OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`

- [AWS Lapisan Lambda terkelola untuk ADOT](#)

2. Siapkan secara manual OpenTelemetry untuk fungsi Lambda Anda:

- Konfigurasikan Prosesor Rentang Sederhana dengan Eksportir Rentang UDP X-Ray
- Siapkan propagator X-Ray Lambda

Membuat data jejak secara manual

Ganti segmen X-Ray dan sub-segmen dengan OpenTelemetry Span:

- Gunakan OpenTelemetry Tracer untuk membuat Span
- Tambahkan atribut ke Span (setara dengan metadata dan anotasi X-Ray)

Important

Saat dikirim ke X-Ray:

- Rentang Server dikonversi ke segmen X-Ray
- Rentang lainnya dikonversi ke sub-segmen X-Ray
- Atribut dikonversi ke metadata secara default

Untuk mengonversi atribut ke anotasi, tambahkan kuncinya ke daftar `aws.xray.annotations` atribut. Untuk informasi selengkapnya, lihat [Mengaktifkan Anotasi X-Ray yang Disesuaikan](#).

Migrasi dari X-Ray Daemon ke AWS CloudWatch agen atau kolektor OpenTelemetry

Anda dapat menggunakan CloudWatch agen atau OpenTelemetry kolektor untuk menerima jejak dari aplikasi instrumentasi Anda dan mengirimkannya ke X-Ray.

Note

CloudWatch Agen versi 1.300025.0 dan yang lebih baru dapat mengumpulkan jejak. OpenTelemetry Menggunakan CloudWatch agen alih-alih Daemon X-Ray mengurangi jumlah agen yang perlu Anda kelola. Untuk informasi selengkapnya, lihat [Mengumpulkan metrik, log, dan jejak dengan CloudWatch agen](#).

Bagian-bagian

- [Migrasi di Amazon EC2 atau server lokal](#)
- [Migrasi di Amazon ECS](#)
- [Migrasi pada Elastic Beanstalk](#)

Migrasi di Amazon EC2 atau server lokal

Important

Hentikan proses Daemon X-Ray sebelum menggunakan CloudWatch agen atau OpenTelemetry kolektor untuk mencegah konflik port.

Pengaturan Daemon X-Ray yang ada

Memasang daemon

Penggunaan Daemon X-Ray Anda yang ada telah diinstal menggunakan salah satu metode berikut:

Instalasi manual

Unduh dan jalankan file yang dapat dieksekusi dari bucket X-Ray daemon Amazon S3.

Instalasi otomatis

Gunakan skrip ini untuk menginstal daemon saat meluncurkan instance:

```
#!/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
```

Mengonfigurasi daemon

Penggunaan Daemon X-Ray Anda yang ada telah dikonfigurasi menggunakan:

- Argumen baris perintah
- File konfigurasi (`xray-daemon.yaml`)

Example Menggunakan file konfigurasi

```
./xray -c ~/xray-daemon.yaml
```

Menjalankan daemon

Penggunaan Daemon X-Ray Anda sudah dimulai dengan perintah berikut:

```
~/xray-daemon$ ./xray -o -n us-east-1
```

Menghapus daemon

Untuk menghapus Daemon X-Ray dari instans Amazon EC2 Anda:

1. Hentikan layanan daemon:

```
systemctl stop xray
```

2. Hapus file konfigurasi:

```
rm ~/path/to/xray-daemon.yaml
```

3. Jika dikonfigurasi, hapus file log:

 Note

Lokasi file log tergantung pada konfigurasi Anda:

- Konfigurasi baris perintah: `/var/log/xray-daemon.log`
- File konfigurasi: Periksa `LogPath` pengaturan

Menyiapkan CloudWatch agen

Memasang agen

Untuk petunjuk penginstalan, lihat [Menginstal CloudWatch agen di server lokal](#).

Mengkonfigurasi agen

1. Buat file konfigurasi untuk mengaktifkan koleksi jejak. Untuk informasi selengkapnya, lihat [Membuat file konfigurasi CloudWatch agen](#).
2. Siapkan izin IAM:
 - Lampirkan peran IAM atau tentukan kredensial untuk agen. Untuk informasi selengkapnya, lihat [Menyiapkan peran IAM](#).
 - Pastikan peran atau kredensialnya menyertakan izin. `xray:PutTraceSegments`

Memulai agen

Untuk instruksi untuk memulai agen, lihat [Memulai CloudWatch agen menggunakan baris perintah](#).

Menyiapkan OpenTelemetry kolektor

Memasang kolektor

Unduh dan instal OpenTelemetry kolektor untuk sistem operasi Anda. Untuk instruksi, lihat [Memasang kolektor](#).

Mengkonfigurasi kolektor

Konfigurasi komponen berikut di kolektor Anda:

- ekstensi `awsproxy`

Diperlukan untuk pengambilan sampel X-Ray

- OTel penerima

Mengumpulkan jejak dari aplikasi Anda

- eksportir xray

Mengirim jejak ke X-Ray

Example Konfigurasi kolektor sampel - otel-collector-config .yaml

```
extensions:
  awsproxy:
    endpoint: 127.0.0.1:2000
  health_check:

receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 127.0.0.1:4317
      http:
        endpoint: 127.0.0.1:4318

processors:
  batch:

exporters:
  awsxray:
    region: 'us-east-1'

service:
  pipelines:
    traces:
      receivers: [otlp]
      exporters: [awsxray]
  extensions: [awsproxy, health_check]
```

⚠ Important

Konfigurasi AWS kredensial dengan izin. `xray:PutTraceSegments` Untuk informasi selengkapnya, lihat [Menentukan kredensial](#).

Memulai kolektor

Jalankan kolektor dengan file konfigurasi Anda:

```
otelcol --config=otel-collector-config.yaml
```

Migrasi di Amazon ECS

⚠ Important

Peran tugas Anda harus memiliki `xray:PutTraceSegments` izin untuk kolektor yang Anda gunakan.

Hentikan kontainer X-Ray Daemon yang ada sebelum menjalankan CloudWatch agen atau wadah OpenTelemetry kolektor pada host yang sama untuk mencegah konflik port.

Menggunakan CloudWatch agen

1. Dapatkan gambar Docker dari Galeri [Publik Amazon ECR](#).
2. Buat file konfigurasi bernama `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. Simpan konfigurasi di Systems Manager Parameter Store:

1. Buka <https://console.aws.amazon.com/systems-manager/>

2. Pilih Buat parameter

3. Masukkan nilai berikut:

- Nama: /ecs/cwagent/otel-config
- Tingkat: Standar
- Tipe: String
- Tipe data: Teks
- Nilai: [Tempel cw-agent-otel konfigurasi.json di sini]

4. Buat definisi tugas menggunakan mode jaringan jembatan:

Dalam ketentuan tugas Anda, konfigurasi tergantung pada mode jaringan yang Anda gunakan. Jaringan jembatan adalah default dan dapat digunakan di VPC default Anda. Dalam jaringan jembatan, atur variabel `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` lingkungan untuk memberi tahu OpenTelemetry SDK apa titik akhir dan port untuk agen. CloudWatch Anda juga harus membuat link dari wadah aplikasi Anda ke wadah Collector untuk jejak yang akan dikirim dari OpenTelemetry SDK dalam aplikasi Anda ke wadah Collector.

Example CloudWatch definisi tugas agen

```
{  
  "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"
      }
    ]
  }
]
}

```

Untuk informasi selengkapnya, lihat [Menerapkan CloudWatch agen untuk mengumpulkan EC2 metrik tingkat instans Amazon di Amazon ECS](#).

Menggunakan OpenTelemetry kolektor

1. Dapatkan gambar Docker otel/opentelemetry-collector-contrib dari [Docker Hub](#).
2. Buat file konfigurasi yang disebut otel-collector-config.yaml menggunakan konten yang sama seperti yang ditunjukkan di Amazon yang EC2 mengonfigurasi bagian kolektor, tetapi perbarui titik akhir untuk digunakan, 0.0.0.0 bukan. 127.0.0.1
3. Untuk menggunakan konfigurasi ini di Amazon ECS, Anda dapat menyimpan konfigurasi di Systems Manager Parameter Store. Pertama, buka konsol Systems Manager Parameter Store, dan pilih Create new parameter. Buat parameter baru dengan informasi berikut:

- Nama: /ecs/otel/config(nama ini akan direferensikan dalam Definisi Tugas untuk Kolektor)
 - Tingkat: Standar
 - Tipe: String
 - Tipe data: Teks
 - Nilai: [Tempel otel-collector-config konfigurasi.yaml di sini]
4. Buat definisi tugas untuk menyebarkan OpenTelemetry kolektor menggunakan mode jaringan jembatan sebagai contoh.

Dalam definisi tugas, konfigurasi tergantung pada mode jaringan yang Anda gunakan. Jaringan jembatan adalah default dan dapat digunakan di VPC default Anda. Dalam jaringan jembatan, atur variabel `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` lingkungan untuk memberi tahu OpenTelemetry SDK apa titik akhir dan port untuk Kolektor. OpenTelemetry Anda juga harus membuat link dari wadah aplikasi Anda ke wadah Collector untuk jejak yang akan dikirim dari OpenTelemetry SDK dalam aplikasi Anda ke wadah Collector.

Example OpenTelemetry definisi tugas kolektor

```
{
  "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"
                }
            ]
        }
    ]
}

```

Migrasi pada Elastic Beanstalk

Important

Hentikan proses Daemon X-Ray sebelum menggunakan CloudWatch agen untuk mencegah konflik port.

Integrasi X-Ray Daemon Anda yang ada diaktifkan dengan menggunakan konsol Elastic Beanstalk, atau dengan mengonfigurasi X-Ray Daemon dalam kode sumber aplikasi Anda dengan file konfigurasi.

Menggunakan CloudWatch agen

Pada platform Amazon Linux 2, konfigurasi CloudWatch agen menggunakan file `.ebextensions` konfigurasi:

1. Buat direktori bernama `.ebextensions` di root proyek Anda
2. Buat file bernama `cloudwatch.config` dalam `.ebextensions` direktori dengan konten berikut:

```
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. Sertakan `.ebextensions` direktori dalam bundel sumber aplikasi Anda saat Anda menerapkan

Untuk informasi selengkapnya tentang file konfigurasi Elastic Beanstalk, [lihat Kustomisasi lingkungan lanjutan](#) dengan file konfigurasi.

Migrasi ke OpenTelemetry Java

Bagian ini memberikan panduan tentang migrasi dari X-Ray SDK ke OpenTelemetry SDK for Java.

Bagian-bagian

- [Solusi instrumentasi otomatis kode nol](#)
- [Solusi instrumentasi manual dengan SDK](#)
- [Menelusuri permintaan yang masuk \(instrumentasi kerangka pegas\)](#)
- [AWS Instrumentasi SDK v2](#)
- [Instrumentasi panggilan HTTP keluar](#)
- [Dukungan instrumentasi untuk perpustakaan lain](#)

- [Membuat data jejak secara manual](#)
- [Instrumentasi Lambda](#)

Solusi instrumentasi otomatis kode nol

With X-Ray Java agent

Untuk mengaktifkan agen Java X-Ray, argumen JVM aplikasi Anda harus dimodifikasi.

```
-javaagent:./path-to-disco/disco-java-agent.jar=pluginPath=./path-to-disco/disco-plugins
```

With OpenTelemetry-based Java agent

Untuk menggunakan agen Java OpenTelemetry berbasis.

- Gunakan AWS Distro for OpenTelemetry (ADOT) Auto-Instrumentation Java agent untuk instrumentasi otomatis dengan agen ADOT Java. Untuk informasi selengkapnya, lihat [Instrumentasi Otomatis untuk Jejak dan Metrik dengan agen Java](#). Jika Anda hanya ingin melacak, nonaktifkan variabel `OTEL_METRICS_EXPORTER=none` lingkungan. untuk mengekspor metrik dari agen Java.

(Opsional) Anda juga dapat mengaktifkan Sinyal CloudWatch Aplikasi saat secara otomatis menginstrumentasi aplikasi Anda AWS dengan instrumentasi otomatis ADOT Java untuk memantau kesehatan aplikasi saat ini dan melacak kinerja aplikasi jangka panjang. Application Signals menyediakan tampilan aplikasi yang terpadu dan berpusat pada aplikasi, layanan, dan dependensi Anda, serta membantu memantau dan memonitor kesehatan aplikasi. Untuk informasi selengkapnya, silakan lihat [Sinyal Aplikasi](#).

- Gunakan agen OpenTelemetry Java untuk instrumentasi otomatis. Untuk informasi selengkapnya, lihat [instrumentasi kode nol dengan Agen Java](#).

Solusi instrumentasi manual dengan SDK

Tracing setup with X-Ray SDK

Untuk menginstruksikan kode Anda dengan X-Ray SDK for Java, pertama, `AWSXRay` kelas harus dikonfigurasi dengan plug-in layanan dan aturan pengambilan sampel lokal, kemudian perekam yang disediakan digunakan.

```
static { AWS XRayRecorderBuilder builder = AWS
XRayRecorderBuilder.standard().withPlugin(new EC2Plugin()).withPlugin(new
ECSPPlugin()); AWS XRay.setGlobalRecorder(builder.build());
}
```

Tracing setup with OpenTelemetry SDK

Dependensi berikut diperlukan.

```
<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>

```

Konfigurasi OpenTelemetry SDK dengan membuat instance `TracerProvider` dan mendaftarkan objek secara global. OpenTelemetrySdk Konfigurasi komponen-komponen ini:

- Eksportir Rentang OTLP (misalnya, `OtlpGrpcSpanExporter`) - Diperlukan untuk mengeksport jejak ke agen atau Kolektor CloudWatch OpenTelemetry
- Propagator AWS X-Ray — Diperlukan untuk menyebarkan Konteks Jejak ke AWS Layanan yang terintegrasi dengan X-Ray
- Sampler Jarak Jauh AWS X-Ray - Diperlukan jika Anda perlu mengambil sampel permintaan menggunakan Aturan Pengambilan Sampel Sinar-X
- Detektor Sumber Daya (misalnya, `EcsResource` atau `Ec2Resource`) - Deteksi metadata host yang menjalankan aplikasi Anda

```

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();
```

Menelusuri permintaan yang masuk (instrumentasi kerangka pegas)

With X-Ray SDK

Untuk informasi tentang cara menggunakan X-Ray SDK dengan framework pegas untuk instrumen aplikasi Anda, lihat [AOP dengan Spring dan X-Ray SDK](#) for Java. Untuk mengaktifkan AOP di Musim Semi, selesaikan langkah-langkah ini.

1. [Konfigurasi Musim Semi](#)
2. [Menambahkan filter penelusuran ke aplikasi Anda](#)
3. [Beri anotasi kode Anda atau terapkan antarmuka](#)
4. [Aktifkan X-Ray di aplikasi Anda](#)

With OpenTelemetry SDK

OpenTelemetry menyediakan pustaka instrumentasi untuk mengumpulkan jejak permintaan masuk untuk aplikasi Spring Boot. Untuk mengaktifkan instrumentasi Spring Boot dengan konfigurasi minimal, sertakan ketergantungan berikut.

```
<dependency>
  <groupId>io.opentelemetry.instrumentation</groupId>
  <artifactId>opentelemetry-spring-boot-starter</artifactId>
</dependency>
```

Untuk informasi selengkapnya tentang cara mengaktifkan dan mengonfigurasi instrumentasi Spring Boot untuk OpenTelemetry penyiapan Anda, lihat OpenTelemetry [Memulai](#).

Using OpenTelemetry-based Java agents

Metode default yang direkomendasikan untuk menginstrumentasi aplikasi Spring Boot adalah dengan menggunakan [agen OpenTelemetry Java](#) dengan instrumentasi bytecode, yang juga menyediakan lebih banyak out-of-the-box instrumentasi dan konfigurasi jika dibandingkan dengan langsung menggunakan SDK. Untuk memulai, lihat [Solusi instrumentasi otomatis kode nol](#).

AWS Instrumentasi SDK v2

With X-Ray SDK

X-Ray SDK for Java dapat secara otomatis AWS menginstruksikan semua klien SDK v2 saat Anda menambahkan sub-modul `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` dalam build Anda.

Untuk menginstrumentasikan panggilan klien hilir klien individual ke AWS layanan dengan AWS SDK for Java 2.2 dan yang lebih baru, `aws-xray-recorder-sdk-aws-sdk-v2-instrumentor` modul dari konfigurasi build Anda dikecualikan dan `aws-xray-recorder-sdk-aws-sdk-v2` modul disertakan. Klien individu diinstrumentasi dengan mengonfigurasinya dengan `a.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

Untuk menginstruksikan semua klien AWS SDK secara otomatis, tambahkan `opentelemetry-aws-sdk-2.2-autoconfigure` sub-modul.

```
<dependency>  
    <groupId>io.opentelemetry.instrumentation</groupId>  
    <artifactId>opentelemetry-aws-sdk-2.2-autoconfigure</artifactId>  
    <version>2.15.0-alpha</version>  
    <scope>runtime</scope>  
</dependency>
```

Untuk menginstrumentasikan klien AWS SDK individual, tambahkan `opentelemetry-aws-sdk-2.2` sub-modul.

```
<dependency>  
    <groupId>io.opentelemetry.instrumentation</groupId>  
    <artifactId>opentelemetry-aws-sdk-2.2</artifactId>  
    <version>2.15.0-alpha</version>  
    <scope>compile</scope>  
</dependency>
```

Kemudian, daftarkan pencegat saat membuat Klien AWS SDK.

```
import io.opentelemetry.instrumentation.awssdk.v2_2.AwsSdkTelemetry;  
  
// ...  
  
    AwsSdkTelemetry telemetry = AwsSdkTelemetry.create(openTelemetry);
```

```
private final S3Client S3_CLIENT = S3Client.builder()
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .addExecutionInterceptor(telemetry.newExecutionInterceptor())
        .build())
    .build();
```

Instrumentasi panggilan HTTP keluar

With X-Ray SDK

Untuk instrumen permintaan HTTP keluar dengan X-Ray, X-Ray SDK untuk Apache HttpClient versi Java diperlukan.

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
```

With OpenTelemetry SDK

Demikian pula dengan X-Ray Java SDK, OpenTelemetry menyediakan `ApacheHttpClientTelemetry` kelas yang memiliki metode pembangun yang memungkinkan pembuatan instance dari sebuah `HttpClientBuilder` untuk menyediakan rentang OpenTelemetry berbasis dan propagasi konteks untuk 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>
```

Berikut ini adalah contoh kode dari [opentelemetry-java-instrumentation](#). Klien HTTP yang disediakan oleh `newHttpClient()` akan menghasilkan jejak untuk permintaan yang dieksekusi.

```
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();
    }
}
```

Dukungan instrumentasi untuk perpustakaan lain

Temukan daftar lengkap instrumentasi Library yang didukung untuk OpenTelemetry Java di GitHub repositori instrumentasi masing-masing, di bawah [Pustaka yang didukung, kerangka kerja, server aplikasi](#), dan. JVMs

Atau, Anda dapat mencari OpenTelemetry Registry untuk mengetahui apakah OpenTelemetry mendukung instrumentasi. Untuk mulai mencari, lihat [Registry](#).

Membuat data jejak secara manual

With X-Ray SDK

Dengan X-Ray SDK, `beginSubsegment` metode `beginSegment` dan metode diperlukan untuk secara manual membuat segmen X-Ray dan sub-segmen.

```
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

Anda dapat menggunakan rentang kustom untuk memantau kinerja aktivitas internal yang tidak ditangkap oleh pustaka instrumentasi. Perhatikan bahwa hanya server jenis rentang yang diubah menjadi segmen X-Ray, semua rentang lainnya diubah menjadi sub-segmen X-Ray.

Pertama, Anda perlu membuat Tracer untuk menghasilkan bentang, yang dapat Anda peroleh melalui metode `iniopenTelemetry.getTracer`. Ini akan memberikan instance Tracer dari `TracerProvider` yang terdaftar secara global dalam [Solusi instrumentasi manual dengan SDK](#) contoh. Anda dapat membuat instance Tracer sebanyak yang diperlukan, tetapi biasanya memiliki satu Tracer untuk seluruh aplikasi.

```
Tracer tracer = openTelemetry.getTracer("my-app");
```

Anda dapat menggunakan Tracer untuk membuat bentang.

```

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();

```

Rentang memiliki tipe internal default.

```

// Default span of type INTERNAL will become an X-Ray subsegment
Span span = tracer.spanBuilder("process-header")
    .startSpan();
try (Scope ignored = span.makeCurrent()) {
    doProcessHeader();
}

```

Menambahkan anotasi dan metadata ke jejak dengan SDK OpenTelemetry

Dalam contoh di atas, `setAttribute` metode ini digunakan untuk menambahkan atribut ke setiap rentang. Secara default, semua atribut rentang akan diubah menjadi metadata dalam data mentah X-Ray. Untuk memastikan bahwa atribut diubah menjadi anotasi dan bukan metadata, contoh di atas menambahkan kunci atribut tersebut ke daftar atribut `aws.xray.annotations`. Untuk informasi selengkapnya, lihat [Mengaktifkan Anotasi dan Anotasi X-Ray yang Disesuaikan](#) dan [metadata](#).

Dengan agen Java OpenTelemetry berbasis

Jika Anda menggunakan agen Java untuk secara otomatis instrumen aplikasi Anda, Anda perlu melakukan instrumentasi manual dalam aplikasi Anda. Misalnya, ke kode instrumen dalam aplikasi untuk bagian yang tidak dicakup oleh pustaka instrumentasi otomatis apa pun.

Untuk melakukan instrumentasi manual dengan agen, Anda perlu menggunakan `opentelemetry-api` artefak. Versi artefak tidak bisa lebih baru dari versi agen.

```
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();
```

Instrumentasi Lambda

With X-Ray SDK

Menggunakan X-Ray SDK, setelah Lambda Anda mengaktifkan Active Tracing, tidak ada konfigurasi tambahan yang diperlukan untuk menggunakan X-Ray SDK. Lambda akan membuat segmen yang mewakili pemanggilan penanganan Lambda, dan Anda dapat membuat sub-segmen atau pustaka instrumen menggunakan X-Ray SDK tanpa konfigurasi tambahan.

With OpenTelemetry-based solutions

Lapisan Lambda instrumentasi otomatis - Anda dapat secara otomatis menginstrumentasi Lambda Anda dengan lapisan Lambda yang dijual menggunakan solusi berikut AWS :

- CloudWatch Sinyal Aplikasi Lapisan Lambda (Disarankan)

Note

Lapisan Lambda ini memiliki Sinyal CloudWatch Aplikasi diaktifkan secara default, yang memungkinkan pemantauan kinerja dan kesehatan untuk aplikasi Lambda Anda dengan mengumpulkan metrik dan jejak. Untuk hanya melacak, atur variabel lingkungan Lambda. `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`

- Mengaktifkan pemantauan kinerja dan kesehatan untuk aplikasi Lambda Anda

- Mengumpulkan metrik dan jejak secara default
- AWS lapisan Lambda dikelola untuk ADOT Java. Untuk informasi selengkapnya, lihat [AWS Distro untuk OpenTelemetry Lambda Support For](#) Java.

Untuk menggunakan instrumentasi manual bersama dengan lapisan instrumentasi otomatis, lihat. [Solusi instrumentasi manual dengan SDK](#) Untuk mengurangi start dingin, pertimbangkan untuk menggunakan instrumentasi OpenTelemetry manual untuk menghasilkan OpenTelemetry jejak untuk fungsi Lambda Anda.

OpenTelemetry instrumentasi manual untuk AWS Lambda

Pertimbangkan kode fungsi Lambda berikut yang membuat panggilan 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);
    }
}

```

Berikut adalah dependensinya.

```

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')
}

```

Untuk menginstruksikan handler Lambda Anda dan klien Amazon S3 secara manual, lakukan hal berikut.

1. Ganti kelas fungsi Anda yang mengimplementasikan `RequestHandler` (atau `RequestStreamHandler`) dengan kelas yang memperpanjang `TracingRequestHandler` (atau `TracingRequestStreamHandler`).
2. Buat instance `TracerProvider` dan daftarkan objek secara global. `OpenTelemetrySdk` `TracerProvider` Disarankan untuk dikonfigurasi dengan:
 - a. Prosesor Rentang Sederhana dengan eksportir rentang UDP X-Ray untuk mengirim Jejak ke titik akhir X-Ray UDP Lambda
 - b. Sampler yang `ParentBased` selalu aktif (Default jika tidak dikonfigurasi)
 - c. Resource dengan `service.name` disetel ke nama fungsi Lambda
 - d. Penyebar X-Ray Lambda
3. Ubah `handleRequest` metode ke `doHandleRequest` dan berikan `OpenTelemetrySdk` objek ke kelas dasar.
4. Instrumentasi klien Amazon S3 dengan instrumentasi `OpenTemetry AWS SDK` dengan mendaftarkan pencegat saat membangun klien.

Anda memerlukan dependensi OpenTelemetry terkait berikut.

```
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')
}
```

Kode berikut menunjukkan fungsi Lambda setelah perubahan yang diperlukan. Anda dapat membuat rentang kustom tambahan untuk melengkapi rentang yang disediakan secara otomatis.

```
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 {

```

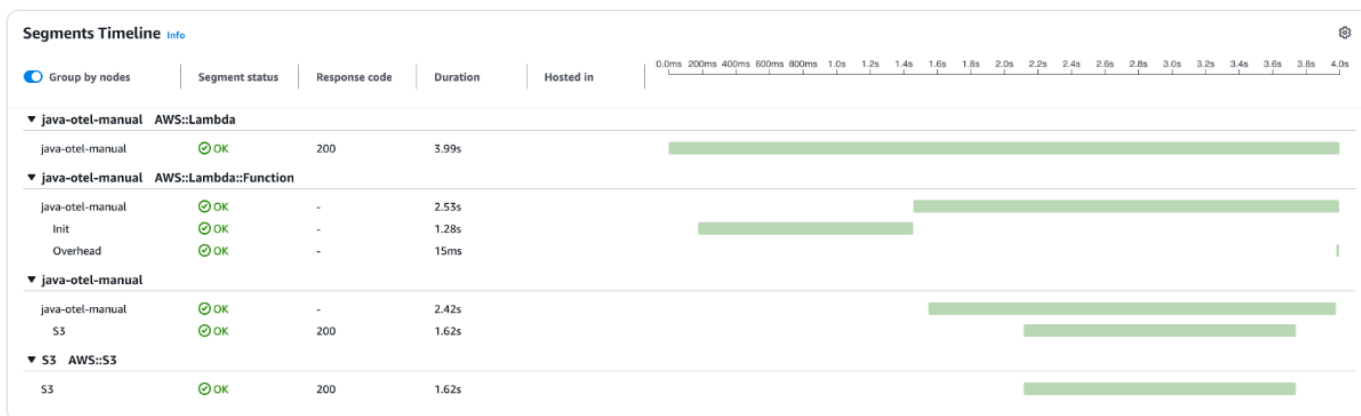
```

        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);
    }
}
}

```

Saat menjalankan fungsi Lambda, Anda akan melihat jejak berikut di bawah Trace Map di CloudWatch konsol.



Migrasi ke Go OpenTelemetry

Gunakan contoh kode berikut untuk mengarahkan aplikasi Go Anda secara manual dengan OpenTelemetry SDK saat bermigrasi dari X-Ray.

Instrumentasi manual dengan SDK

Tracing setup with X-Ray SDK

Saat menggunakan X-Ray SDK for Go, plugin layanan atau aturan pengambilan sampel lokal harus dikonfigurasi sebelum menginstrumentasi kode Anda.

```
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

Konfigurasi OpenTelemetry SDK dengan membuat instance TracerProvider dan mendaftarkannya sebagai penyedia pelacak global. Kami merekomendasikan untuk mengonfigurasi komponen-komponen berikut:

- OTLP Trace Exporter - Diperlukan untuk mengeksport jejak ke Agen atau Kolektor CloudWatch OpenTelemetry
- X-Ray Propagator — Diperlukan untuk menyebarkan konteks jejak ke AWS layanan yang terintegrasi dengan X-Ray
- X-Ray Remote Sampler — Diperlukan untuk permintaan sampling menggunakan aturan sampling X-Ray
- Detektor sumber daya - Untuk mendeteksi metadata host yang menjalankan aplikasi Anda

```
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
}
```

Menelusuri permintaan yang masuk (instrumentasi penanganan HTTP)

With X-Ray SDK

Untuk instrumen handler HTTP dengan X-Ray, metode handler X-Ray digunakan untuk menghasilkan segmen menggunakan `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

Untuk instrumen handler HTTP dengan OpenTelemetry, gunakan metode `NewHandler` untuk membungkus kode handler asli Anda. OpenTelemetry

```
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 Instrumentasi SDK for Go v2

With X-Ray SDK

Untuk menginstrumentasi AWS permintaan keluar dari AWS SDK, klien Anda diinstrumentasi sebagai berikut:

```
// 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.AWSSV2Instrumentor(&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

Menelusuri dukungan untuk panggilan AWS SDK hilir disediakan oleh SDK for AWS Go OpenTelemetry v2 Instrumentation. Berikut adalah contoh penelusuran panggilan klien 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
}

```

Instrumentasi panggilan HTTP keluar

With X-Ray SDK

Untuk instrumen panggilan HTTP keluar dengan X-Ray, `Xray.client` digunakan untuk membuat salinan klien HTTP yang disediakan.

```
myClient := xray.Client(http-client)

resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

With OpenTelemetry SDK

Untuk instrumen klien HTTP dengan OpenTelemetry, gunakan OpenTelemetry otelhttp. NewTransport metode untuk membungkus 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)
```

Dukungan instrumentasi untuk perpustakaan lain

Anda dapat menemukan daftar lengkap instrumentasi pustaka yang didukung untuk OpenTelemetry Go di bawah paket [Instrumentasi](#).

Atau, Anda dapat mencari OpenTelemetry registri untuk mengetahui apakah OpenTelemetry mendukung instrumentasi untuk perpustakaan Anda di bawah [Registry](#).

Membuat data jejak secara manual

With X-Ray SDK

Dengan X-Ray SDK, `BeginSubsegment` metode `BeginSegment` dan diperlukan untuk membuat segmen dan sub-segmen X-Ray secara manual.

```
// 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

Gunakan rentang kustom untuk memantau kinerja aktivitas internal yang tidak ditangkap oleh pustaka instrumentasi. Perhatikan bahwa hanya rentang jenis `Server` yang diubah menjadi segmen X-Ray, semua rentang lainnya diubah menjadi sub-segmen X-Ray.

Pertama, Anda perlu membuat `Tracer` untuk menghasilkan bentang, yang dapat Anda peroleh melalui metode `initotel.Tracer`. Ini akan memberikan instance `Tracer` dari `TracerProvider` yang terdaftar secara global dalam contoh `Tracing Setup`. Anda dapat membuat instance `Tracer` sebanyak yang diperlukan, tetapi biasanya memiliki satu `Tracer` untuk seluruh aplikasi.

```
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()
```

Menambahkan anotasi dan metadata ke jejak dengan SDK OpenTelemetry

Dalam contoh di atas, `WithAttributes` metode ini digunakan untuk menambahkan atribut ke setiap rentang. Perhatikan bahwa secara default, semua atribut rentang diubah menjadi metadata dalam data mentah X-Ray. Untuk memastikan bahwa atribut diubah menjadi anotasi dan bukan metadata, tambahkan kunci atribut ke daftar atribut. `aws.xray.annotations` Untuk informasi selengkapnya, lihat [Mengaktifkan Anotasi X-Ray yang Disesuaikan](#).

Instrumentasi manual Lambda

With X-Ray SDK

Dengan X-Ray SDK, setelah Lambda Anda mengaktifkan Active Tracing, tidak ada konfigurasi tambahan yang diperlukan untuk menggunakan X-Ray SDK. Lambda membuat segmen yang mewakili pemanggilan penanganan Lambda, dan Anda membuat sub-segmen menggunakan X-Ray SDK tanpa konfigurasi tambahan.

With OpenTelemetry SDK

Kode fungsi Lambda berikut (tanpa instrumentasi) membuat ListBuckets panggilan Amazon S3 dan permintaan HTTP keluar.

```
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)  
}
```

Untuk menginstruksikan handler Lambda Anda dan klien Amazon S3 secara manual, lakukan hal berikut:

1. Di main (), buat instance a TracerProvider (tp) dan daftarkan sebagai penyedia pelacak global. TracerProvider Disarankan untuk dikonfigurasi dengan:
 - a. Prosesor Rentang Sederhana dengan eksportir rentang UDP X-Ray untuk mengirim Jejak ke titik akhir X-Ray UDP Lambda
 - b. Sumber daya dengan service.name disetel ke nama fungsi Lambda
2. Ubah penggunaan `lambda.Start(lambdaHandler)` ke `lambda.Start(otelambda.InstrumentHandler(lambdaHandler, xrayconfig.WithRecommendedOptions(tp)...))`.
3. Instrumentasikan klien Amazon S3 dengan instrumentasi OpenTelemetry AWS SDK dengan menambahkan OpenTelemetry middleware ke dalam `aws-sdk-go-v2` konfigurasi klien Amazon S3.
4. Instrumen klien http dengan menggunakan OpenTelemetry `otelhttp.NewTransport` metode untuk membungkus `http.DefaultTransport`.

Kode berikut adalah contoh bagaimana Fungsi Lambda akan terlihat seperti setelah perubahan. Anda dapat secara manual membuat rentang kustom tambahan selain rentang yang disediakan secara otomatis.

```
package main  
  
import (  
    "context"  
    "encoding/json"  
    "fmt"  
    "io"  
    "net/http"  
    "os"  
  
    "github.com/aws-observability/aws-otel-go/exporters/xrayudp"  
    "github.com/aws/aws-lambda-go/events"
```

```

    "github.com/aws/aws-lambda-go/lambda"
    awsconfig "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"

    lambdadetector "go.opentelemetry.io/contrib/detectors/aws/lambda"
    "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-lambda-go/
otellambda"
    "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-lambda-go/
otellambda/xrayconfig"
    "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-sdk-go-v2/
otelaws"
    "go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp"
    "go.opentelemetry.io/contrib/propagators/aws/xray"
    "go.opentelemetry.io/otel"
    "go.opentelemetry.io/otel/attribute"
    "go.opentelemetry.io/otel/sdk/resource"
    "go.opentelemetry.io/otel/sdk/trace"
    semconv "go.opentelemetry.io/otel/semconv/v1.26.0"
)

func lambdaHandler(ctx context.Context) (interface{}, error) {
    // Initialize AWS config.
    cfg, err := awsconfig.LoadDefaultConfig(ctx)
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    // Instrument all AWS clients.
    otelaws.AppendMiddlewares(&cfg.APIOptions)
    // Create an instrumented S3 client from the config.
    s3Client := s3.NewFromConfig(cfg)

    // Create an instrumented HTTP client.
    httpClient := &http.Client{
        Transport: otelhttp.NewTransport(
            http.DefaultTransport,
        ),
    }

    // return func(ctx context.Context) (interface{}, error) {
    input := &s3.ListBucketsInput{}
    result, err := s3Client.ListBuckets(ctx, input)
    if err != nil {
        fmt.Printf("Got an error retrieving buckets, %v", err)
    }
}

```

```

    }

    fmt.Println("Buckets:")
    for _, bucket := range result.Buckets {
        fmt.Println(*bucket.Name + ": " + bucket.CreationDate.Format("2006-01-02
15:04:05 Monday"))
    }
    fmt.Println("End Buckets.")

    req, err := http.NewRequestWithContext(ctx, http.MethodGet, "https://
api.github.com/repos/aws-observability/aws-otel-go/releases/latest", nil)
    if err != nil {
        fmt.Printf("failed to create http request, %v\n", err)
    }
    res, err := httpClient.Do(req)
    if err != nil {
        fmt.Printf("failed to make http request, %v\n", err)
    }
    defer func(Body io.ReadCloser) {
        err := Body.Close()
        if err != nil {
            fmt.Printf("failed to close http response body, %v\n", err)
        }
    }(res.Body)

    var data map[string]interface{}
    err = json.NewDecoder(res.Body).Decode(&data)
    if err != nil {
        fmt.Printf("failed to read http response body, %v\n", err)
    }
    fmt.Printf("Latest ADOT Go Release is '%s'\n", data["name"])

    return events.APIGatewayProxyResponse{
        StatusCode: http.StatusOK,
        Body:       os.Getenv("_X_AMZN_TRACE_ID"),
    }, nil
}

func main() {
    ctx := context.Background()
    detector := lambdadetector.NewResourceDetector()
    lambdaResource, err := detector.Detect(context.Background())
    if err != nil {
        fmt.Printf("failed to detect lambda resources: %v\n", err)
    }
}

```

```
}

var attributes = []attribute.KeyValue{
    attribute.KeyValue{Key: semconv.ServiceNameKey, Value:
attribute.StringValue(os.Getenv("AWS_LAMBDA_FUNCTION_NAME"))},
}
customResource := resource.NewWithAttributes(semconv.SchemaURL, attributes...)
mergedResource, _ := resource.Merge(lambdaResource, customResource)

xrayUdpExporter, _ := xrayudp.NewSpanExporter(ctx)
tp := trace.NewTracerProvider(
    trace.WithSpanProcessor(trace.NewSimpleSpanProcessor(xrayUdpExporter)),
    trace.WithResource(mergedResource),
)

defer func(ctx context.Context) {
    err := tp.Shutdown(ctx)
    if err != nil {
        fmt.Printf("error shutting down tracer provider: %v", err)
    }
}(ctx)

otel.SetTracerProvider(tp)
otel.SetTextMapPropagator(xray.Propagator{})

lambda.Start(otelambda.InstrumentHandler(lambdaHandler,
xrayconfig.WithRecommendedOptions(tp)...))
}
```

Saat memanggil Lambda, Anda akan melihat jejak berikut Trace Map di CloudWatch konsol:



Migrasi ke Node.js OpenTelemetry

Bagian ini menjelaskan cara memigrasikan aplikasi Node.js Anda dari X-Ray SDK ke. OpenTelemetry. Ini mencakup pendekatan instrumentasi otomatis dan manual, dan memberikan contoh spesifik untuk kasus penggunaan umum.

X-Ray Node.js SDK membantu Anda secara manual instrumen aplikasi Node.js Anda untuk melacak. Bagian ini memberikan contoh kode untuk migrasi dari X-Ray ke OpenTelemetry instrumentasi.

Bagian-bagian

- [Solusi instrumentasi otomatis kode nol](#)
- [Solusi instrumentasi manual](#)
- [Menelusuri permintaan yang masuk](#)
- [AWS Instrumentasi SDK JavaScript V3](#)
- [Instrumentasi panggilan HTTP keluar](#)
- [Dukungan instrumentasi untuk perpustakaan lain](#)
- [Membuat data jejak secara manual](#)
- [Instrumentasi Lambda](#)

Solusi instrumentasi otomatis kode nol

Untuk melacak permintaan dengan X-Ray SDK untuk Node.js, Anda harus memodifikasi kode aplikasi Anda. Dengan OpenTelemetry, Anda dapat menggunakan solusi instrumentasi otomatis kode nol untuk melacak permintaan.

Instrumentasi otomatis kode nol dengan instrumentasi otomatis OpenTelemetry berbasis.

1. Menggunakan instrumentasi otomatis AWS Distro for OpenTelemetry (ADOT) untuk Node.js — Untuk instrumentasi otomatis untuk aplikasi Node.js, lihat [Penelusuran dan Metrik dengan Distro untuk Instrumentasi Otomatis](#). AWS OpenTelemetry JavaScript

(Opsional) Anda juga dapat mengaktifkan Sinyal CloudWatch Aplikasi saat secara otomatis menginstrumentasi aplikasi Anda AWS dengan JavaScript instrumentasi otomatis ADOT untuk memantau kesehatan aplikasi saat ini dan melacak kinerja aplikasi jangka panjang terhadap tujuan bisnis Anda. Sinyal Aplikasi memberi Anda tampilan menyatu dan terpusat aplikasi pada aplikasi, layanan, dan dependensi Anda, serta membantu Anda memantau dan menentukan prioritas kesehatan aplikasi. Untuk informasi selengkapnya, silakan lihat [Sinyal Aplikasi](#).

2. [Menggunakan instrumentasi otomatis OpenTelemetry JavaScript kode nol — Untuk instrumentasi otomatis dengan, lihat instrumentasi kode nol. OpenTelemetry JavaScript JavaScript](#)

Solusi instrumentasi manual

Tracing setup with X-Ray SDK

Ketika X-Ray SDK untuk Node.js digunakan, `aws-xray-sdk` paket diperlukan untuk mengonfigurasi X-Ray SDK dengan plug-in layanan atau aturan pengambilan sampel lokal sebelum menggunakan SDK untuk instrumen kode Anda.

```
var AWSXRay = require('aws-xray-sdk');

AWSXRay.config([AWSXRay.plugins.EC2Plugin, AWSXRay.plugins.ElasticBeanstalkPlugin]);
AWSXRay.middleware.setSamplingRules(<path to file>);
```


Tracing setup with OpenTelemetry SDK

Note

AWS X-Ray Remote Sampling saat ini tidak tersedia untuk dikonfigurasi untuk OpenTelemetry JS. Namun, dukungan untuk X-Ray Remote Sampling saat ini tersedia melalui ADOT Auto-Instrumentation untuk Node.js.

Untuk contoh kode di bawah ini, Anda memerlukan dependensi berikut:

```
npm install --save \  
  @opentelemetry/api \  
  @opentelemetry/sdk-node \  
  @opentelemetry/exporter-trace-otlp-proto \  
  @opentelemetry/propagator-aws-xray \  
  @opentelemetry/resource-detector-aws
```

Anda harus mengatur dan mengkonfigurasi OpenTelemetry SDK sebelum menjalankan kode aplikasi Anda. Ini dapat dilakukan dengan menggunakan tanda [—require](#). Buat file bernama `instrumentation.js`, yang akan berisi konfigurasi dan pengaturan OpenTelemetry instrumentasi Anda.

Disarankan agar Anda mengkonfigurasi komponen-komponen berikut:

- OTLPTraceEksportir - Diperlukan untuk mengeksport jejak ke Agen/Kolektor CloudWatch OpenTelemetry
- AWSXRayPropagator — Diperlukan untuk menyebarkan Konteks Jejak ke AWS Layanan yang terintegrasi dengan X-Ray
- Detektor Sumber Daya (misalnya, Amazon EC2 Resource Detector) - Untuk mendeteksi metadata host yang menjalankan aplikasi Anda

```
/*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();
```

Kemudian, Anda dapat menjalankan aplikasi Anda dengan OpenTelemetry pengaturan Anda seperti:

```
node --require ./instrumentation.js app.js
```

Anda dapat menggunakan instrumentasi pustaka OpenTelemetry SDK untuk membuat rentang pustaka secara otomatis seperti SDK. AWS Mengaktifkan ini akan secara otomatis membuat rentang untuk modul seperti AWS SDK untuk v3. JavaScript OpenTelemetry menyediakan opsi untuk mengaktifkan semua instrumentasi perpustakaan atau menentukan instrumentasi perpustakaan mana yang akan diaktifkan.

Untuk mengaktifkan semua instrumentasi, instal paket: @opentelemetry/auto-instrumentations-node

```
npm install @opentelemetry/auto-instrumentations-node
```

Selanjutnya, perbarui konfigurasi untuk mengaktifkan semua instrumentasi perpustakaan seperti yang ditunjukkan di bawah ini.

```
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

Anda dapat menggunakan instrumentasi otomatis ADOT untuk Node.js untuk mengkonfigurasi OpenTelemetry secara otomatis untuk aplikasi Node.js Anda. Dengan menggunakan ADOT Auto-Instrumentation, Anda tidak perlu membuat perubahan kode manual untuk melacak permintaan yang masuk, atau untuk melacak pustaka seperti klien SDK atau HTTP. AWS Untuk informasi selengkapnya, lihat [Menelusuri dan metrik dengan AWS Distro untuk](#) Instrumentasi Otomatis. OpenTelemetry JavaScript

Instrumentasi otomatis ADOT untuk Node.js mendukung:

- Pengambilan sampel jarak jauh X-Ray melalui variabel lingkungan - `export OTEL_TRACES_SAMPLER=xray`
- Propagasi konteks jejak X-Ray (diaktifkan secara default)
- Deteksi sumber daya (deteksi sumber daya untuk lingkungan Amazon EC2, Amazon ECS, dan Amazon EKS diaktifkan secara default)
- Instrumentasi pustaka otomatis untuk semua OpenTelemetry instrumentasi yang didukung, yang dapat disabled/enabled selektif melalui dan variabel lingkungan `OTEL_NODE_ENABLED_INSTRUMENTATIONS` `OTEL_NODE_DISABLED_INSTRUMENTATIONS`
- Pembuatan manual Spans

Menelusuri permintaan yang masuk

With X-Ray SDK

Express.js

Dengan X-Ray SDK untuk melacak permintaan HTTP masuk yang diterima oleh aplikasi Express.js, kedua middlewares `AWSXRay.express.openSegment(<name>)` dan `AWSXRay.express.closeSegment()` diminta untuk membungkus semua rute yang Anda tentukan untuk melacaknya.

```
app.use(xrayExpress.openSegment('defaultName'));  
  
...  
  
app.use(xrayExpress.closeSegment());
```

Membantah

Untuk melacak permintaan HTTP masuk yang diterima oleh Restify aplikasi, middleware dari X-Ray SDK digunakan dengan menjalankan `enable` dari `aws-xray-sdk-restify` modul di 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

[Menelusuri dukungan untuk permintaan masuk Express.js disediakan oleh instrumentasi OpenTelemetry HTTP dan OpenTelemetry instrumentasi ekspres.](#) Instal dependensi berikut dengan: `npm`

```
npm install --save @opentelemetry/instrumentation-http @opentelemetry/instrumentation-express
```

Perbarui Konfigurasi OpenTelemetry SDK untuk mengaktifkan instrumentasi untuk modul ekspres:

```
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(),
  ],
});
```

Membantah

Untuk aplikasi Restify, Anda memerlukan instrumentasi [OpenTelemetry Restify](#). Instal ketergantungan berikut:

```
npm install --save @opentelemetry/instrumentation-restify
```

Perbarui Konfigurasi OpenTelemetry SDK untuk mengaktifkan instrumentasi untuk modul restify:

```
const { RestifyInstrumentation } = require('@opentelemetry/instrumentation-restify');
...
```

```
const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new RestifyInstrumentation(),
  ],
});
```

AWS Instrumentasi SDK JavaScript V3

With X-Ray SDK

Untuk menginstrumentasi AWS permintaan keluar dari AWS SDK, Anda menginstrumentasi klien seperti contoh berikut:

```
import { S3, PutObjectCommand } from '@aws-sdk/client-s3';

const s3 = AWSXRay.captureAWSSv3Client(new S3({}));

await s3.send(new PutObjectCommand({
  Bucket: bucketName,
  Key: keyName,
  Body: 'Hello!',
}));
```

With OpenTelemetry SDK

Menelusuri dukungan untuk panggilan AWS SDK hilir ke DynamoDB, Amazon S3, dan lainnya disediakan oleh instrumentasi SDK. OpenTelemetry AWS Instal ketergantungan berikut dengannpm:

```
npm install --save @opentelemetry/instrumentation-aws-sdk
```

Perbarui Konfigurasi OpenTelemetry SDK dengan instrumentasi AWS SDK.

```
import { AwsInstrumentation } from '@opentelemetry/instrumentation-aws-sdk';
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new AwsInstrumentation()
  ],
});
```

Instrumentasi panggilan HTTP keluar

With X-Ray SDK

Untuk instrumen permintaan HTTP keluar dengan X-Ray, itu diperlukan untuk instrumen klien. Misalnya, lihat di bawah.

Klien HTTP individu

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

Semua klien HTTP (global)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

With OpenTelemetry SDK

Menelusuri dukungan untuk klien HTTP Node.js disediakan oleh Instrumentasi OpenTelemetry HTTP. Instal ketergantungan berikut dengannpm:

```
npm install --save @opentelemetry/instrumentation-http
```

Perbarui Konfigurasi OpenTelemetry SDK sebagai berikut:

```
const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
...

const sdk = new NodeSDK({
  ...

  instrumentations: [
    ...
    new HttpInstrumentation(),
  ],
});
```

Dukungan instrumentasi untuk perpustakaan lain

Anda dapat menemukan daftar lengkap instrumentasi pustaka yang didukung OpenTelemetry JavaScript di bawah Instrumentasi yang [didukung](#).

Atau, Anda dapat mencari OpenTelemetry registri untuk mengetahui apakah OpenTelemetry mendukung instrumentasi untuk perpustakaan Anda di bawah [Registry](#).

Membuat data jejak secara manual

With X-Ray SDK

Menggunakan X-Ray, kode `aws-xray-sdk` paket diperlukan untuk secara manual membuat segmen dan sub-segmen anak mereka untuk melacak aplikasi Anda.

```
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

Anda dapat membuat dan menggunakan rentang kustom untuk memantau kinerja aktivitas internal yang tidak ditangkap oleh pustaka instrumentasi. Perhatikan bahwa hanya rentang jenis Server yang diubah menjadi segmen X-Ray, semua rentang lainnya diubah menjadi sub-segmen X-Ray. Untuk informasi selengkapnya, lihat [Segmen](#).

Anda akan memerlukan instance Tracer setelah mengonfigurasi OpenTelemetry SDK di Tracing Setup untuk membuat Span. Anda dapat membuat instance Tracer sebanyak yang diperlukan, tetapi biasanya memiliki satu Tracer untuk seluruh aplikasi.

```
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();
});
```

Menambahkan anotasi dan metadata ke jejak dengan SDK OpenTelemetry

Anda juga dapat menambahkan pasangan nilai kunci kustom sebagai atribut dalam rentang Anda. Perhatikan bahwa secara default, semua atribut rentang ini akan diubah menjadi metadata dalam data mentah X-Ray. Untuk memastikan bahwa atribut diubah menjadi anotasi dan bukan metadata, tambahkan kunci atribut ke daftar atribut. `aws.xray.annotations` Untuk informasi selengkapnya, lihat [Mengaktifkan Anotasi X-Ray yang Disesuaikan](#).

```
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();
});
```

Instrumentasi Lambda

With X-Ray SDK

Setelah Anda mengaktifkan Active Tracing untuk fungsi Lambda Anda, X-Ray SDK diperlukan tanpa konfigurasi tambahan. Lambda membuat segmen yang mewakili pemanggilan penanganan Lambda, dan Anda membuat sub-segmen atau pustaka instrumen menggunakan X-Ray SDK tanpa konfigurasi tambahan apa pun.

With OpenTelemetry SDK

Anda dapat secara otomatis menginstruksikan Lambda Anda dengan lapisan Lambda AWS yang dijual. Ada dua solusi:

- (Disarankan) Sinyal CloudWatch Aplikasi lapisan lambda

Note

Lapisan Lambda ini memiliki Sinyal CloudWatch Aplikasi diaktifkan secara default, yang memungkinkan pemantauan kinerja dan kesehatan untuk aplikasi Lambda Anda dengan mengumpulkan metrik dan jejak. Jika Anda hanya ingin melacak, Anda harus mengatur variabel lingkungan Lambda. `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false` Untuk informasi selengkapnya, lihat [Mengaktifkan aplikasi Anda di Lambda](#).

- AWS lapisan Lambda terkelola untuk ADOT JS. Untuk informasi selengkapnya, lihat [AWS Distro untuk OpenTelemetry Lambda Support JavaScript For](#).

Membuat Span secara manual dengan instrumentasi Lambda

Sementara ADOT JavaScript Lambda Layer menyediakan instrumentasi otomatis untuk fungsi Lambda Anda, Anda mungkin menemukan kebutuhan untuk melakukan instrumentasi manual di Lambda Anda, misalnya, untuk menyediakan data kustom atau kode instrumen dalam fungsi Lambda itu sendiri yang tidak tercakup oleh instrumentasi perpustakaan.

Untuk melakukan instrumentasi manual bersama instrumentasi otomatis, Anda perlu menambahkan `@opentelemetry/api` sebagai ketergantungan. Versi dependensi ini direkomendasikan untuk menjadi versi yang sama dari dependensi yang sama yang digunakan oleh JavaScript ADOT SDK. Anda dapat menggunakan OpenTelemetry API untuk membuat bentang secara manual di fungsi Lambda Anda.

Untuk menambahkan `@opentelemetry/api` ketergantungan menggunakan NPM:

```
npm install @opentelemetry/api
```

Migrasi ke OpenTelemetry .NET

Saat menggunakan X-Ray Tracing di aplikasi.NET Anda, X-Ray .NET SDK dengan upaya manual digunakan untuk instrumentasi.

Bagian ini memberikan contoh kode di [Solusi instrumentasi manual dengan SDK](#) bagian untuk bermigrasi dari solusi instrumentasi manual X-Ray ke solusi Instrumentasi OpenTelemetry manual untuk.NET. Atau, Anda dapat bermigrasi dari instrumentasi manual X-Ray ke solusi instrumentasi OpenTelemetry otomatis ke instrumen aplikasi.NET tanpa harus memodifikasi kode sumber aplikasi di bagian ini. [Solusi instrumentasi otomatis kode nol](#)

Bagian-bagian

- [Solusi instrumentasi otomatis kode nol](#)
- [Solusi instrumentasi manual dengan SDK](#)
- [Membuat data jejak secara manual](#)
- [Menelusuri permintaan yang masuk \(instrumentasi inti ASP.NET dan ASP.NET\)](#)
- [AWS Instrumentasi SDK](#)
- [Instrumentasi panggilan HTTP keluar](#)
- [Dukungan instrumentasi untuk perpustakaan lain](#)
- [Instrumentasi Lambda](#)

Solusi instrumentasi otomatis kode nol

OpenTelemetry menyediakan solusi instrumentasi otomatis kode nol. Solusi ini melacak permintaan tanpa memerlukan perubahan pada kode aplikasi Anda.

OpenTelemetryopsi instrumentasi otomatis berbasis

1. Menggunakan AWS Distro for OpenTelemetry (ADOT) Auto-instrumentasi untuk.NET — Untuk secara otomatis menginstrumentasikan aplikasi.NET, lihat [Tracing and Metrics with the Distro for .NET Auto-Instrumentation](#). AWS OpenTelemetry

(Opsional) Aktifkan Sinyal CloudWatch Aplikasi saat secara otomatis menginstrumentasi aplikasi Anda AWS dengan instrumentasi otomatis ADOT.NET ke:

- Pantau kesehatan aplikasi saat ini
- Melacak kinerja aplikasi jangka panjang terhadap tujuan bisnis
- Dapatkan tampilan terpadu, aplikasi-sentris dari aplikasi, layanan, dan dependensi Anda
- Monitor dan triase aplikasi kesehatan

Untuk informasi selengkapnya, silakan lihat [Sinyal Aplikasi](#).

2. Menggunakan instrumentasi OpenTelemetry otomatis. Net zero-code — Untuk instrumen otomatis dengan .Net secara otomatis, lihat [Tracing dan Metrik dengan Distro for OpenTelemetry .NET Auto-Instrumentation](#). AWS OpenTelemetry

Solusi instrumentasi manual dengan SDK

Tracing configuration with X-Ray SDK

Untuk aplikasi web.NET, X-Ray SDK dikonfigurasi di bagian AppSettings file. `Web.config`

Contoh Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
  </appSettings>
</configuration>
```

Untuk .NET Core, file bernama `appsettings.json` dengan kunci tingkat atas bernama `XRay` digunakan, dan kemudian objek konfigurasi dibangun untuk menginisialisasi perekam X-Ray.

Contoh untuk .NET appsettings.json

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin"
  }
}
```

Contoh untuk .NET Core Program.cs - Konfigurasi perekam

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

Tracing configuration with OpenTelemetry SDK

Tambahkan dependensi ini:

```
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
```

Untuk aplikasi.NET Anda, konfigurasi OpenTelemetry SDK dengan menyiapkan Global TracerProvider. Contoh konfigurasi berikut juga memungkinkan instrumentasi untuk ASP.NET Core. Untuk instrumen ASP.NET, lihat [Menelusuri permintaan yang masuk \(instrumentasi inti ASP.NET dan ASP.NET\)](#). Untuk digunakan OpenTelemetry dengan kerangka kerja lain, lihat [Registry](#) untuk lebih banyak pustaka untuk kerangka kerja yang didukung.

Disarankan agar Anda mengkonfigurasi komponen-komponen berikut:

- An OTLP Exporter- Diperlukan untuk mengeksport jejak ke CloudWatch OpenTelemetry Agen/Kolektor
- Propagator AWS X-Ray — Diperlukan untuk menyebarkan Konteks Jejak ke [AWS Layanan yang terintegrasi dengan X-Ray](#)
- Sampler Jarak Jauh AWS X-Ray - Diperlukan jika Anda perlu [mengambil sampel permintaan menggunakan aturan pengambilan sampel X-Ray](#)
- Resource Detectors (misalnya, Amazon EC2 Resource Detector) - Untuk mendeteksi metadata host yang menjalankan aplikasi Anda

```
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
```

Untuk digunakan OpenTelemetry untuk aplikasi konsol, tambahkan OpenTelemetry konfigurasi berikut saat memulai program Anda.

```
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

```

Membuat data jejak secara manual

With X-Ray SDK

Dengan X-Ray SDK, `BeginSubsegment` metode `BeginSegment` dan diperlukan untuk membuat segmen dan sub-segmen X-Ray secara manual.

```

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

Di .NET, Anda dapat menggunakan API aktivitas untuk membuat rentang kustom guna memantau kinerja aktivitas internal yang tidak ditangkap oleh pustaka instrumentasi. Perhatikan bahwa hanya rentang jenis Server yang diubah menjadi segmen X-Ray, semua rentang lainnya diubah menjadi sub-segment X-Ray.

Anda dapat membuat `ActivitySource` instance sebanyak yang diperlukan, tetapi disarankan untuk hanya memiliki satu untuk seluruh aplikasi/layanan.

```
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
    }
}

```

Menambahkan anotasi dan metadata ke jejak dengan SDK OpenTelemetry

Anda juga dapat menambahkan pasangan nilai kunci kustom sebagai atribut ke rentang Anda dengan menggunakan `SetTag` metode pada aktivitas. Perhatikan bahwa secara default, semua atribut rentang akan diubah menjadi metadata dalam data mentah X-Ray. Untuk memastikan bahwa atribut diubah menjadi anotasi dan bukan metadata, Anda dapat menambahkan kunci atribut tersebut ke daftar atribut. `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
    }
}

```

Dengan instrumentasi OpenTelemetry otomatis

Jika Anda menggunakan solusi instrumentasi OpenTelemetry otomatis untuk .NET, dan jika Anda perlu melakukan instrumentasi manual dalam aplikasi Anda, misalnya, ke kode instrumen dalam aplikasi itu sendiri untuk bagian yang tidak tercakup oleh pustaka instrumentasi otomatis apa pun.

Karena hanya ada satu `globalTracerProvider`, instrumentasi manual tidak boleh membuat instance sendiri `TracerProvider` jika digunakan bersama dengan instrumentasi otomatis. Saat `TracerProvider` digunakan, penelusuran manual khusus bekerja dengan cara yang sama saat menggunakan instrumentasi otomatis atau instrumentasi manual melalui SDK. `OpenTelemetry`

Menelusuri permintaan yang masuk (instrumentasi inti ASP.NET dan ASP.NET)

With X-Ray SDK

Untuk permintaan instrumen yang disajikan oleh aplikasi ASP.NET, lihat <https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-dotnet-messagehandler.html> informasi tentang cara menelepon `RegisterXRay` dalam `Init` metode `global.asax` file Anda.

```
AWSXRayAspNet.RegisterXRay(this, "MyApp");
```

Untuk permintaan instrumen yang disajikan oleh aplikasi inti ASP.NET Anda, `UseXRay` metode ini dipanggil sebelum middleware lain dalam `Configure` metode kelas `Startup` Anda.

```
app.UseXRay("MyApp");
```

With OpenTelemetry SDK

`OpenTelemetry` juga menyediakan pustaka instrumentasi untuk mengumpulkan jejak permintaan web yang masuk untuk ASP.NET dan ASP.NET core. Bagian berikut mencantumkan langkah-langkah yang diperlukan untuk menambahkan dan mengaktifkan instrumentasi pustaka ini untuk `OpenTelemetry` konfigurasi Anda, termasuk cara menambahkan instrumentasi inti [ASP.NET](#) atau [ASP.NET](#) saat membuat `Penyedia Pelacak`.

Untuk informasi tentang cara mengaktifkan `OpenTelemetry .Instrumentation. AspNet`, lihat [Langkah-langkah untuk mengaktifkan OpenTelemetry .Instrumentation. AspNet](#) dan untuk informasi tentang cara mengaktifkan `OpenTelemetry .Instrumentation. AspNetCore`, lihat [Langkah-langkah untuk mengaktifkan OpenTelemetry .Instrumentation. AspNetCore](#).

AWS Instrumentasi SDK

With X-Ray SDK

Instal semua klien AWS SDK dengan menelepon `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
```

Gunakan salah satu metode berikut untuk instrumentasi klien AWS layanan tertentu.

```
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

Untuk contoh kode berikut, Anda memerlukan ketergantungan berikut:

```
dotnet add package OpenTelemetry.Instrumentation.AWS
```

Untuk AWS menginstrumentasikan SDK, perbarui konfigurasi OpenTelemetry SDK tempat `GlobalTracerProvider` disiapkan.

```
builder.Services.AddOpenTelemetry()
    ...
    .WithTracing(tracing => tracing
        .AddAWSInstrumentation()
        ...
```

Instrumentasi panggilan HTTP keluar

With X-Ray SDK

X-Ray .NET SDK melacak panggilan HTTP keluar melalui metode ekstensi `GetResponseTraced()` atau `GetAsyncResponseTraced()` saat menggunakan `System.Net.HttpWebRequest`, atau dengan menggunakan `HttpClientXRayTracingHandler` handler saat menggunakan `System.Net.Http.HttpClient`.

With OpenTelemetry SDK

Untuk contoh kode berikut, Anda memerlukan ketergantungan berikut:

```
dotnet add package OpenTelemetry.Instrumentation.Http
```

Untuk instrumen `System.Net.Http.HttpClient` dan `System.Net.HttpWebRequest`, perbarui konfigurasi OpenTelemetry SDK tempat `Global TracerProvider` diatur.

```
builder.Services.AddOpenTelemetry()  
    ...  
    .WithTracing(tracing => tracing  
        .AddHttpClientInstrumentation()  
        ...
```

Dukungan instrumentasi untuk perpustakaan lain

Anda dapat mencari dan memfilter OpenTelemetry Registry untuk .NET Instrumentation Libraries untuk mengetahui apakah OpenTelemetry mendukung instrumentasi untuk Library Anda. Lihat [Registry](#) untuk mulai mencari.

Instrumentasi Lambda

With X-Ray SDK

Prosedur berikut diperlukan untuk menggunakan X-Ray SDK dengan Lambda:

1. Aktifkan Penelusuran Aktif pada fungsi Lambda Anda
2. Layanan Lambda membuat segmen yang mewakili pemanggilan handler Anda
3. Membuat sub-segmen atau pustaka instrumen menggunakan X-Ray SDK

With OpenTelemetry-based solutions

Anda dapat secara otomatis menginstruksikan Lambda Anda dengan lapisan Lambda AWS yang dijual. Ada dua solusi:

- (Disarankan) [Sinyal CloudWatch Aplikasi lapisan lambda](#)
- Untuk kinerja yang lebih baik, Anda mungkin ingin mempertimbangkan untuk menggunakan OpenTelemetry Manual Instrumentation untuk menghasilkan OpenTelemetry jejak untuk fungsi Lambda Anda.

OpenTelemetry instrumentasi manual untuk AWS Lambda

Berikut ini adalah contoh kode fungsi Lambda (tanpa instrumentasi).

```
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);
    }
}
}
```

Untuk menginstruksikan handler Lambda Anda dan klien Amazon S3 secara manual, lakukan hal berikut.

1. Instantiate a TracerProvider - TracerProvider Disarankan untuk dikonfigurasi dengan `XrayUdpSpanExporter`, `ParentBased Always On Sampler`, dan `Resource` dengan `service.name` set ke nama fungsi Lambda.
2. Instrumentasi klien Amazon S3 dengan instrumentasi OpenTemetry AWS SDK dengan menelepon `AddAWSInstrumentation()` untuk menambahkan AWS instrumentasi klien SDK `TracerProvider`
3. Buat fungsi pembungkus dengan tanda tangan yang sama dengan fungsi Lambda asli. Panggil `AWSLambdaWrapper.Trace()` API dan `lulusTracerProvider`, fungsi Lambda asli, dan inputnya sebagai parameter. Atur fungsi pembungkus sebagai input handler Lambda.

Untuk contoh kode berikut, Anda memerlukan dependensi berikut:

```
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
```

Kode berikut menunjukkan fungsi Lambda setelah perubahan yang diperlukan. Anda dapat membuat rentang kustom tambahan untuk melengkapi rentang yang disediakan secara otomatis.

```
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);
}
}
```

Saat menjalankan Lambda ini, Anda akan melihat jejak berikut di Peta Jejak di CloudWatch konsol:



Migrasi ke Python OpenTelemetry

Panduan ini membantu Anda memigrasikan aplikasi Python dari X-Ray SDK OpenTelemetry ke instrumentasi. Ini mencakup pendekatan instrumentasi otomatis dan manual, dengan contoh kode untuk skenario umum.

Bagian-bagian

- [Solusi instrumentasi otomatis kode nol](#)
- [Instrumen aplikasi Anda secara manual](#)
- [Menelusuri inisialisasi pengaturan](#)
- [Menelusuri permintaan yang masuk](#)
- [AWS Instrumentasi SDK](#)
- [Menginstrumentasi panggilan HTTP keluar melalui permintaan](#)
- [Dukungan instrumentasi untuk perpustakaan lain](#)

- [Membuat data jejak secara manual](#)
- [Instrumentasi Lambda](#)

Solusi instrumentasi otomatis kode nol

Dengan X-Ray SDK, Anda harus memodifikasi kode aplikasi Anda untuk melacak permintaan. OpenTelemetry menawarkan solusi instrumentasi otomatis kode nol untuk melacak permintaan. Dengan OpenTelemetry, Anda memiliki opsi untuk menggunakan solusi instrumentasi otomatis kode nol untuk melacak permintaan.

Kode nol dengan instrumentasi otomatis OpenTelemetry berbasis

1. Menggunakan AWS Distro for OpenTelemetry (ADOT) Auto-instrumentation for Python — Untuk instrumentasi otomatis untuk aplikasi Python, lihat [Tracing](#) and Metrics with the Distro for Python Auto-Instrumentation. AWS OpenTelemetry

(Opsional) Anda juga dapat mengaktifkan Sinyal CloudWatch Aplikasi saat secara otomatis menginstrumentasi aplikasi Anda AWS dengan instrumentasi otomatis ADOT Python untuk memantau kesehatan aplikasi saat ini dan melacak kinerja aplikasi jangka panjang terhadap tujuan bisnis Anda. Sinyal Aplikasi memberi Anda tampilan menyatu dan terpusat aplikasi pada aplikasi, layanan, dan dependensi Anda, serta membantu Anda memantau dan menentukan prioritas kesehatan aplikasi.

2. [Menggunakan instrumentasi otomatis kode nol OpenTelemetry Python — Untuk instrumentasi otomatis dengan Python, lihat instrumentasi kode nol Python. OpenTelemetry](#)

Instrumen aplikasi Anda secara manual

Anda dapat secara manual instrumen aplikasi Anda menggunakan pip perintah.

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
```

Menelusuri inisialisasi pengaturan

With X-Ray SDK

Dalam X-Ray, global `xray_recorder` diinisialisasi dan digunakan untuk menghasilkan segmen dan sub-segmen.

With OpenTelemetry SDK

Note

X-Ray Remote Sampling saat ini tidak tersedia untuk dikonfigurasi untuk OpenTelemetry Python. Namun, dukungan untuk X-Ray Remote Sampling saat ini tersedia melalui ADOT Auto-Instrumentation untuk Python.

Di OpenTelemetry, Anda perlu menginisialisasi `globalTracerProvider`. Dengan menggunakan `iniTracerProvider`, Anda dapat memperoleh [Tracer](#) yang dapat Anda gunakan untuk menghasilkan rentang di mana saja di aplikasi Anda. Disarankan agar Anda mengkonfigurasi komponen-komponen berikut:

- `OTLPSpanExporter`- Diperlukan untuk mengeksport jejak ke CloudWatch OpenTelemetry Agen/Kolektor
- `Propagator AWS X-Ray` — Diperlukan untuk menyebarkan Konteks Jejak ke AWS Layanan yang terintegrasi dengan 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

```

Dengan ADOT Auto-instrumentasi untuk Python

Anda dapat menggunakan instrumentasi otomatis ADOT untuk Python untuk mengkonfigurasi secara otomatis untuk aplikasi Python Anda. OpenTelemetry Dengan menggunakan instrumentasi otomatis ADOT, Anda tidak perlu membuat perubahan kode manual untuk melacak permintaan yang masuk, atau untuk melacak pustaka seperti klien SDK atau HTTP. AWS Untuk informasi selengkapnya, lihat [Menelusuri dan Metrik dengan AWS Distro untuk Instrumentasi Otomatis Python OpenTelemetry](#).

Instrumentasi otomatis ADOT untuk Python mendukung:

- X-Ray Remote Sampling melalui variabel lingkungan `export OTEL_TRACES_SAMPLER=xray`
- Propagasi konteks jejak X-Ray (diaktifkan secara default)
- Deteksi sumber daya (deteksi sumber daya untuk lingkungan Amazon EC2, Amazon ECS, dan Amazon EKS diaktifkan secara default)
- Instrumentasi pustaka otomatis untuk semua OpenTelemetry instrumentasi yang didukung diaktifkan secara default. Anda dapat menonaktifkan secara selektif melalui variabel `OTEL_PYTHON_DISABLED_INSTRUMENTATIONS` lingkungan. (semua diaktifkan secara default)

- Pembuatan manual Spans

Dari plug-in layanan X-Ray hingga penyedia sumber daya OpenTelemetry AWS

X-Ray SDK menyediakan plug-in yang dapat Anda tambahkan ke untuk menangkap informasi spesifik platform dari layanan yang dihosting seperti Amazon, EC2 Amazon ECS, dan Elastic Beanstalk. `xray_recorder` Ini mirip dengan Penyedia Sumber Daya OpenTelemetry yang menangkap informasi sebagai atribut Sumber Daya. Ada beberapa Penyedia Sumber Daya yang tersedia untuk berbagai AWS platform.

- Mulailah dengan menginstal paket AWS ekstensi, `pip install opentelemetry-sdk-extension-aws`
- Konfigurasi detektor sumber daya yang diinginkan. Contoh berikut menunjukkan cara mengonfigurasi penyedia EC2 sumber daya Amazon di OpenTelemetry SDK

```
from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.extension.aws.resource.ec2 import (
    AwsEc2ResourceDetector,
)
from opentelemetry.sdk.resources import get_aggregated_resources

provider = TracerProvider(
    active_span_processor=span_processor,
    resource=get_aggregated_resources([
        AwsEc2ResourceDetector(),
    ]))

trace.set_tracer_provider(provider)
```

Menelusuri permintaan yang masuk

With X-Ray SDK

X-Ray Python SDK mendukung kerangka aplikasi seperti Django, Flask, dan Bottle dalam menelusuri permintaan masuk untuk aplikasi Python yang berjalan pada mereka. Ini dilakukan dengan menambahkan aplikasi `XRayMiddleware` untuk setiap kerangka kerja.

With OpenTelemetry SDK

OpenTelemetry menyediakan instrumentasi untuk [Django](#) dan [Flask](#) melalui pustaka instrumentasi tertentu. Tidak ada instrumentasi untuk Botol yang tersedia di OpenTelemetry, aplikasi masih dapat dilacak dengan menggunakan Instrumentasi [OpenTelemetry WSGI](#).

Untuk contoh kode berikut, Anda memerlukan ketergantungan berikut:

```
pip install opentelemetry-instrumentation-flask
```

Anda harus menginisialisasi OpenTelemetry SDK dan mendaftarkan global TracerProvider sebelum menambahkan instrumentasi untuk kerangka kerja aplikasi Anda. Tanpa itu, operasi jejak akan terdajino-ops. Setelah Anda mengonfigurasi globalTracerProvider, Anda dapat menggunakan instrumentor untuk kerangka kerja aplikasi Anda. Contoh berikut menunjukkan aplikasi 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 Instrumentasi SDK

With X-Ray SDK

X-Ray Python SDK melacak permintaan klien AWS SDK dengan menambal perpustakaan `botocore`. Untuk informasi selengkapnya, lihat [Menelusuri panggilan AWS SDK dengan X-Ray SDK untuk Python](#). Dalam aplikasi Anda, `patch_all()` metode ini digunakan untuk instrumen semua pustaka atau patch selektif menggunakan pustaka atau menggunakan `boto3` pustaka. `botocore patch(['botocore'])` Salah satu metode yang dipilih menginstrumentasikan semua klien Boto3 dalam aplikasi Anda dan menghasilkan sub-segmen untuk setiap panggilan yang dilakukan menggunakan klien ini.

With OpenTelemetry SDK

Untuk contoh kode berikut, Anda memerlukan ketergantungan berikut:

```
pip install opentelemetry-instrumentation-botocore
```

Gunakan [OpenTelemetry Botocore Instrumentation](#) secara terprogram untuk instrumen semua klien Boto3 dalam aplikasi Anda. Contoh berikut menunjukkan `botocore` instrumentasi.

```
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}")
```

Menginstrumentasi panggilan HTTP keluar melalui permintaan

With X-Ray SDK

X-Ray Python SDK melacak panggilan HTTP keluar melalui permintaan dengan menambal perpustakaan permintaan. Untuk informasi selengkapnya, lihat [Menelusuri panggilan ke layanan web HTTP hilir menggunakan X-Ray SDK untuk Python](#). Dalam aplikasi Anda, Anda dapat menggunakan `patch_all()` metode ini untuk menginstrumentasi semua pustaka atau dengan menambal pustaka permintaan secara selektif dengan menggunakan `patch(['requests'])`. Salah satu opsi instrumen `requests` perpustakaan, menghasilkan sub-segmen untuk setiap panggilan yang dilakukan. `requests`

With OpenTelemetry SDK

Untuk contoh kode berikut, Anda memerlukan ketergantungan berikut:

```
pip install opentelemetry-instrumentation-requests
```

Gunakan Instrumentasi OpenTelemetry Permintaan secara terprogram untuk menginstrumentasikan pustaka permintaan untuk menghasilkan jejak permintaan HTTP yang dibuat olehnya dalam aplikasi Anda. Untuk informasi selengkapnya, lihat [OpenTelemetry permintaan Instrumentasi](#). Contoh berikut menunjukkan instrumentasi `requests` perpustakaan.

```
from opentelemetry.instrumentation.requests import RequestsInstrumentor

# Instrument Requests
RequestsInstrumentor().instrument()

...

example_session = requests.Session()
example_session.get(url="https://example.com")
```

Atau, Anda juga dapat menginstruksikan `urllib3` pustaka yang mendasarinya untuk melacak permintaan 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")
```

Dukungan instrumentasi untuk perpustakaan lain

Anda dapat menemukan daftar lengkap instrumentasi Library yang didukung untuk OpenTelemetry Python di bawah [Pustaka, kerangka kerja, server aplikasi, dan file yang didukung](#). JVMs

Atau, Anda dapat mencari OpenTelemetry Registry untuk mengetahui apakah OpenTelemetry mendukung instrumentasi. Lihat [Registry](#) untuk mulai mencari.

Membuat data jejak secara manual

Anda dapat membuat segmen dan sub-segmen menggunakan `xray_recorder` dalam aplikasi Python Anda. Untuk informasi selengkapnya, lihat [Menginstrumentasi kode Python](#) secara manual. Anda juga dapat menambahkan anotasi dan metadata secara manual ke data jejak.

Membuat bentang Dengan OpenTelemetry SDK

Gunakan `start_as_current_span` API untuk memulai rentang dan atur untuk membuat bentang. Untuk contoh cara membuat bentang, lihat [Membuat bentang](#). Setelah rentang dimulai dan berada dalam lingkup saat ini, Anda dapat menambahkan lebih banyak informasi ke dalamnya dengan menambahkan atribut, peristiwa, pengecualian, tautan, dan sebagainya. Seperti bagaimana kita memiliki segmen dan sub-segmen dalam X-Ray, ada berbagai jenis bentang di. OpenTelemetry Hanya rentang SERVER jenis yang diubah menjadi segmen X-Ray sementara yang lain diubah menjadi sub-segmen X-Ray.

```
from opentelemetry import trace
from opentelemetry.trace import SpanKind
```

```
import time

tracer = trace.get_tracer("my.tracer.name")

# Create a new span to track some work
with tracer.start_as_current_span("parent", kind=SpanKind.SERVER) as parent_span:
    time.sleep(1)

    # Create a nested span to track nested work
    with tracer.start_as_current_span("child", kind=SpanKind.CLIENT) as child_span:
        time.sleep(2)
        # the nested span is closed when it's out of scope

    # Now the parent span is the current span again
    time.sleep(1)

# This span is also closed when it goes out of scope
```

Menambahkan anotasi dan metadata ke jejak dengan SDK OpenTelemetry

X-Ray Python SDK menyediakan terpisah APIs, `put_annotation` dan `put_metadata` untuk menambahkan anotasi dan metadata ke jejak. Di OpenTelemetry SDK, anotasi dan metadata hanyalah atribut pada rentang, ditambahkan melalui API. `set_attribute`

Atribut rentang yang Anda inginkan untuk menjadi anotasi pada jejak ditambahkan di bawah kunci cadangan `aws.xray.annotations` yang nilainya adalah daftar pasangan kunci-nilai anotasi. Semua atribut rentang lainnya menjadi metadata pada segmen atau sub-segmen yang dikonversi.

Selain itu, jika Anda menggunakan kolektor ADOT, Anda dapat mengonfigurasi atribut rentang mana yang harus dikonversi ke anotasi X-Ray `indexed_attributes` dengan menentukan konfigurasi kolektor.

Contoh di bawah ini menunjukkan cara menambahkan anotasi dan metadata ke jejak menggunakan 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")
```

Instrumentasi Lambda

Untuk memantau fungsi lambda Anda pada X-Ray, Anda mengaktifkan X-Ray dan menambahkan izin yang sesuai ke peran pemanggilan fungsi. Selain itu, jika Anda melacak permintaan hilir dari fungsi Anda, Anda akan menginstrumentasi kode dengan X-Ray Python SDK.

Dengan OpenTelemetry X-Ray, disarankan untuk menggunakan lapisan lambda Sinyal CloudWatch Aplikasi dengan [Sinyal Aplikasi](#) dimatikan. Ini akan mengotomatiskan fungsi Anda dan akan menghasilkan rentang untuk pemanggilan fungsi dan permintaan hilir apa pun dari fungsi Anda. Selain melacak, jika Anda tertarik menggunakan Sinyal Aplikasi untuk memantau kesehatan fungsi Anda, lihat [Mengaktifkan aplikasi Anda di Lambda](#).

- Temukan ARN layer Lambda yang diperlukan untuk fungsi Anda dari [AWS Lambda](#) Layer dan tambahkan. OpenTelemetry ARNs
- Tetapkan variabel lingkungan berikut untuk fungsi Anda.
 - `AWS_LAMBDA_EXEC_WRAPPER=/opt/otel-instrument`— Ini memuat instrumentasi otomatis untuk fungsi tersebut
 - `OTEL_AWS_APPLICATION_SIGNALS_ENABLED=false`— Ini akan menonaktifkan pemantauan Sinyal Aplikasi

Membuat bintang secara manual dengan instrumentasi Lambda

Selain itu, Anda dapat menghasilkan rentang kustom dalam fungsi Anda untuk melacak pekerjaan. Anda dapat melakukannya dengan hanya menggunakan `opentelemetry-api` paket dalam hubungannya dengan Application Signals lambda layer auto-instrumentation.

1. Sertakan `opentelemetry-api` sebagai dependensi dalam fungsi Anda
2. Cuplikan kode berikut adalah contoh untuk menghasilkan rentang kustom

```
from opentelemetry import trace

# Get the tracer (auto-configured by the Application Signals layer)
tracer = trace.get_tracer(__name__)

def handler(event, context):
    # This span is a child of the layer's root span
    with tracer.start_as_current_span("my-custom-span") as span:
        span.set_attribute("key1", "value1")
        span.add_event("custom-event", {"detail": "something happened"})

        # Any logic you want to trace
        result = some_internal_logic()

    return {
        "statusCode": 200,
        "body": result
    }
```

Migrasi ke Ruby OpenTelemetry

Untuk memigrasikan aplikasi Ruby Anda dari X-Ray SDK ke OpenTelemetry instrumentasi, gunakan contoh kode berikut dan panduan untuk instrumentasi manual.

Bagian-bagian

- [Instrumentasikan solusi Anda secara manual dengan SDK](#)
- [Menelusuri permintaan masuk \(instrumentasi Rails\)](#)
- [AWS Instrumentasi SDK](#)
- [Instrumentasi panggilan HTTP keluar](#)
- [Dukungan instrumentasi untuk perpustakaan lain](#)
- [Membuat data jejak secara manual](#)
- [Instrumentasi manual Lambda](#)

Instrumentasikan solusi Anda secara manual dengan SDK

Tracing setup with X-Ray SDK

X-Ray SDK for Ruby mengharuskan Anda untuk mengonfigurasi kode Anda dengan plug-in layanan.

```
require 'aws-xray-sdk'

XRay.recorder.configure(plugins: [:ec2, :elastic_beanstalk])
```

Tracing setup with OpenTelemetry SDK

Note

X-Ray Remote Sampling saat ini tidak tersedia untuk dikonfigurasi untuk OpenTelemetry Ruby.

Untuk aplikasi Ruby on Rails, tempatkan kode konfigurasi Anda di penginisialisasi Rails. Untuk informasi selengkapnya, lihat [Memulai](#). Untuk semua program Ruby yang diinstrumentasi secara manual, Anda harus menggunakan `OpenTelemetry::SDK.configure` metode ini untuk mengkonfigurasi OpenTelemetry Ruby SDK.

Pertama, instal paket-paket berikut:

```
bundle add opentelemetry-sdk opentelemetry-exporter-otlp opentelemetry-propagator-xray
```

Selanjutnya, konfigurasikan OpenTelemetry SDK melalui kode konfigurasi yang berjalan saat program Anda diinisialisasi. Disarankan agar Anda mengkonfigurasi komponen-komponen berikut:

- `OTLP Exporter`- Diperlukan untuk mengeksport jejak ke CloudWatch agen dan kolektor OpenTelemetry
- `An AWS X-Ray Propagator`— Diperlukan untuk menyebarkan konteks jejak ke AWS layanan yang terintegrasi dengan 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 juga memiliki konsep instrumentasi perpustakaan. Mengaktifkan ini akan secara otomatis membuat rentang untuk pustaka seperti SDK. AWS OpenTelemetry menyediakan opsi untuk mengaktifkan semua instrumentasi perpustakaan atau menentukan instrumentasi perpustakaan mana yang akan diaktifkan.

Untuk mengaktifkan semua instrumentasi, instal paket terlebih dahulu: `opentelemetry-instrumentation-all`

```
bundle add opentelemetry-instrumentation-all
```

Selanjutnya, perbarui konfigurasi untuk mengaktifkan semua instrumentasi pustaka seperti yang ditunjukkan di bawah ini:

```
require 'opentelemetry/instrumentation/all'
```



```
...

OpenTelemetry::SDK.configure do |c|
  ...

  c.use_all() # Enable all instrumentations
end
```

OpenTelemetry SDKs juga memiliki konsep instrumentasi perpustakaan. Mengaktifkan ini akan secara otomatis membuat rentang untuk pustaka seperti SDK. AWS OpenTelemetry menyediakan opsi untuk mengaktifkan semua instrumentasi perpustakaan atau menentukan instrumentasi perpustakaan mana yang akan diaktifkan.

Untuk mengaktifkan semua instrumentasi, instal paket terlebih dahulu: `opentelemetry-instrumentation-all`

```
bundle add opentelemetry-instrumentation-all
```

Selanjutnya, perbarui konfigurasi untuk mengaktifkan semua instrumentasi pustaka seperti yang ditunjukkan di bawah ini:

```
require 'opentelemetry/instrumentation/all'
...

OpenTelemetry::SDK.configure do |c|
  ...

  c.use_all() # Enable all instrumentations
end
```

Menelusuri permintaan masuk (instrumentasi Rails)

With X-Ray SDK

Dengan X-Ray SDK, penelusuran X-Ray dikonfigurasi untuk kerangka Rails setelah inisialisasi.

Contoh - `config/initializers/aws_xray.rb`

```
Rails.application.config.xray = {  
  name: 'my app',  
  patch: %I[net_http aws_sdk],  
  active_record: true  
}
```

With OpenTelemetry SDK

Pertama, instal paket-paket berikut:

```
bundle add opentelemetry-instrumentation-rack opentelemetry-instrumentation-rails opentelemetry-instrumentation-action_pack opentelemetry-instrumentation-active_record opentelemetry-instrumentation-action_view
```

Selanjutnya, perbarui konfigurasi untuk mengaktifkan instrumentasi untuk aplikasi Rails Anda seperti yang ditunjukkan di bawah ini:

```
# 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 Instrumentasi SDK

With X-Ray SDK

Untuk menginstrumentasikan AWS permintaan keluar dari AWS SDK, klien AWS SDK ditambah dengan X-Ray seperti contoh berikut:

```
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 menyediakan dukungan untuk merekam dan memancarkan jejak.

OpenTelemetry Untuk informasi tentang cara mengonfigurasi klien layanan, lihat [Mengonfigurasi fitur observabilitas di AWS SDK OpenTelemetry for Ruby](#).

Instrumentasi panggilan HTTP keluar

Saat melakukan panggilan HTTP ke layanan eksternal, Anda mungkin perlu mengarahkan panggilan secara manual jika instrumentasi otomatis tidak tersedia atau tidak memberikan detail yang cukup.

With X-Ray SDK

Untuk instrumen panggilan hilir, X-Ray SDK for Ruby digunakan untuk menambal `net/http` library yang digunakan aplikasi Anda:

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http]
}

XRay.recorder.configure(config)
```

With OpenTelemetry SDK

Untuk mengaktifkan `net/http` instrumentasi menggunakan OpenTelemetry, pertama instal `opentelemetry-instrumentation-net_http` paket:

```
bundle add opentelemetry-instrumentation-net_http
```

Selanjutnya, perbarui konfigurasi untuk mengaktifkan net/http instrumentasi seperti yang ditunjukkan di bawah ini:

```
OpenTelemetry::SDK.configure do |c|  
  ...  
  
  c.use 'OpenTelemetry::Instrumentation::Net::HTTP'  
  ...  
  
end
```

Dukungan instrumentasi untuk perpustakaan lain

Anda dapat menemukan daftar lengkap instrumentasi Library yang didukung untuk OpenTelemetry Ruby di bawah. [opentelemetry-ruby-contrib](#)

Atau, Anda dapat mencari OpenTelemetry Registry untuk mengetahui apakah OpenTelemetry mendukung instrumentasi. Untuk informasi selengkapnya, lihat [Registry](#).

Membuat data jejak secara manual

With X-Ray SDK

Menggunakan X-Ray, `aws-xray-sdk` paket mengharuskan Anda untuk membuat segmen dan sub-segmen anak mereka secara manual untuk melacak aplikasi Anda. Anda mungkin juga telah menambahkan anotasi X-Ray dan metadata ke segmen atau sub-segmen Anda:

```
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

Anda dapat menggunakan rentang kustom untuk memantau kinerja aktivitas internal yang tidak ditangkap oleh pustaka instrumentasi. Perhatikan bahwa hanya rentang server jenis yang diubah menjadi segmen X-Ray, semua rentang lainnya diubah menjadi sub-segmen X-Ray. Secara default, bentang adalah INTERNAL.

Pertama, buat Tracer untuk menghasilkan bentang, yang dapat Anda peroleh melalui metode `iniOpenTelemetry.tracer_provider.tracer('<YOUR_TRACER_NAME>')`. Ini akan memberikan instance Tracer yang terdaftar secara global dalam OpenTelemetry

konfigurasi aplikasi Anda. Adalah umum untuk memiliki satu Tracer untuk seluruh aplikasi. Buat OpenTelemetry pelacak dan gunakan untuk membuat bentang:

```
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
```

Menambahkan anotasi dan metadata ke jejak dengan SDK OpenTelemetry

Gunakan `set_attribute` metode untuk menambahkan atribut ke setiap rentang. Perhatikan bahwa secara default, semua atribut rentang ini akan diubah menjadi metadata dalam data mentah X-Ray. Untuk memastikan bahwa atribut diubah menjadi anotasi dan bukan metadata, Anda dapat menambahkan kunci atribut tersebut ke daftar atribut `aws.xray.annotations`. Untuk informasi selengkapnya, lihat [Mengaktifkan Anotasi X-Ray yang Disesuaikan](#).

```
# 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")
end
```

```
span.set_attribute("annotationKey1", "annotationValue")

# Create X-Ray annotations
span.set_attribute("aws.xray.annotations", ["annotationKey1"])
end
```

Instrumentasi manual Lambda

With X-Ray SDK

Setelah Active Tracing diaktifkan di Lambda, tidak ada konfigurasi tambahan yang diperlukan untuk menggunakan X-Ray SDK. Lambda akan membuat segmen yang mewakili pemanggilan penanganan Lambda, dan Anda dapat membuat sub-segmen atau pustaka instrumen menggunakan X-Ray SDK tanpa konfigurasi tambahan.

With OpenTelemetry SDK

Pertimbangkan contoh kode fungsi Lambda berikut (tanpa instrumentasi):

```
require 'json'
def lambda_handler(event:, context:)
  # TODO implement
  { statusCode: 200, body: JSON.generate('Hello from Lambda!') }
end
```

Untuk instrumen Lambda Anda secara manual, Anda harus:

1. Tambahkan permata berikut untuk Lambda Anda

```
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. Inisialisasi OpenTelemetry SDK di luar Lambda Handler Anda. OpenTelemetry SDK direkomendasikan untuk dikonfigurasi dengan:

1. Prosesor span sederhana dengan eksportir rentang UDP X-Ray untuk mengirim Jejak ke titik akhir X-Ray UDP Lambda
2. Penyebar X-Ray Lambda
3. `service_name` konfigurasi yang akan diatur ke nama fungsi Lambda
3. Di kelas handler Lambda Anda, tambahkan baris berikut untuk instrumen Lambda Handler Anda:

```
class Handler
  extend OpenTelemetry::Instrumentation::AwsLambda::Wrap
  ...

  instrument_handler :process
end
```

Kode berikut menunjukkan fungsi Lambda setelah perubahan yang diperlukan. Anda dapat membuat rentang kustom tambahan untuk melengkapi rentang yang disediakan secara otomatis.

```
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]

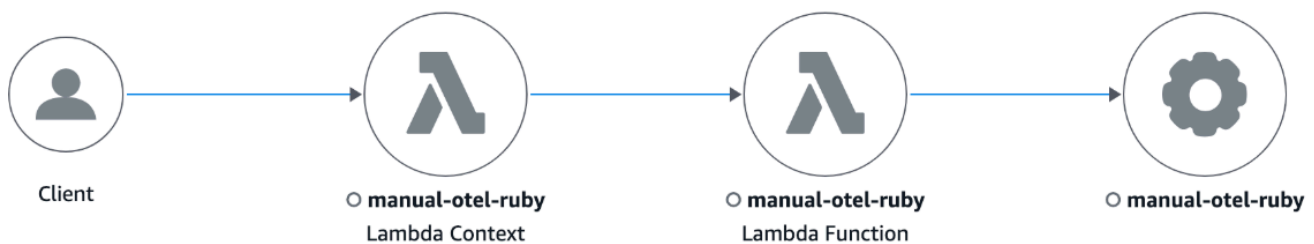
  # Set minimal resource information
  c.resource = OpenTelemetry::SDK::Resources::Resource.create({
    OpenTelemetry::SemanticConventions::Resource::SERVICE_NAME =>
    ENV['AWS_LAMBDA_FUNCTION_NAME']
  })
```



```
})
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
```

Berikut ini adalah contoh peta jejak fungsi Lambda berinstrumen yang ditulis dalam Ruby.



Anda juga dapat menggunakan lapisan Lambda untuk mengkonfigurasi OpenTelemetry Lambda Anda. Untuk informasi selengkapnya, lihat Instrumentasi [OpenTelemetry AWS-Lambda](#).

Membuat sumber daya X-Ray dengan AWS CloudFormation

AWS X-Ray terintegrasi dengan AWS CloudFormation, layanan yang membantu Anda memodelkan dan mengatur AWS sumber daya Anda sehingga Anda dapat menghabiskan lebih sedikit waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat template yang menjelaskan semua AWS sumber daya yang Anda inginkan, dan menyediakan serta AWS CloudFormation mengonfigurasi sumber daya tersebut untuk Anda.

Ketika Anda menggunakan AWS CloudFormation, Anda dapat menggunakan kembali template Anda untuk mengatur sumber daya X-Ray Anda secara konsisten dan berulang kali. Jelaskan sumber daya Anda sekali, lalu sediakan sumber daya yang sama berulang-ulang di beberapa Akun AWS dan Wilayah.

X-Ray dan AWS CloudFormation template

Untuk menyediakan dan mengonfigurasi sumber daya untuk X-Ray dan layanan terkait, Anda harus memahami [templat AWS CloudFormation](#). Templat adalah file teks dengan format JSON atau YAML. Template ini menjelaskan sumber daya yang ingin Anda sediakan di AWS CloudFormation tumpukan Anda. Jika Anda tidak terbiasa dengan JSON atau YAMAL, Anda dapat menggunakan AWS CloudFormation Designer untuk membantu Anda memulai dengan template. AWS CloudFormation Untuk informasi selengkapnya, lihat [Apa itu AWS CloudFormation Designer?](#) di Panduan Pengguna AWS CloudFormation .

X-Ray mendukung pembuatan [AWS::XRay::Group](#), [AWS::XRay::SamplingRule](#), dan [AWS::XRay::ResourcePolicy](#) sumber daya di AWS CloudFormation. Untuk informasi selengkapnya, termasuk contoh templat JSON dan YAML, lihat [referensi tipe sumber daya X-Ray](#) dalam Panduan Pengguna AWS CloudFormation .

Pelajari lebih lanjut tentang AWS CloudFormation

Untuk mempelajari selengkapnya AWS CloudFormation, lihat sumber daya berikut:

- [AWS CloudFormation](#)
- [AWS CloudFormation Panduan Pengguna](#)
- [AWS CloudFormation Referensi API](#)
- [AWS CloudFormation Panduan Pengguna Antarmuka Baris Perintah](#)

Menandai aturan dan grup pengambilan sampel X-Ray

Tag adalah kata atau frasa yang dapat Anda gunakan untuk mengidentifikasi dan mengatur AWS sumber daya Anda. Anda dapat menambahkan beberapa tanda pada setiap sumber daya. Setiap tanda mencakup kunci dan nilai opsional yang Anda tentukan. Misalnya, kunci tanda mungkin **domain**, dan nilai tanda mungkin **example.com**. Anda dapat mencari dan memfilter sumber daya Anda berdasarkan tanda yang Anda tambahkan. Untuk informasi selengkapnya tentang cara menggunakan tanda, lihat [Penandaan sumber daya AWS](#) di Referensi Umum AWS .

Anda dapat menggunakan tag untuk menerapkan izin berbasis tag pada distribusi. CloudFront Untuk informasi selengkapnya, lihat [Mengontrol Akses ke Sumber Daya AWS Menggunakan Tanda Sumber Daya](#).

Note

[Editor Tanda](#) dan [Resource Groups AWS](#) saat ini tidak mendukung sumber daya X-Ray. Anda menambahkan dan mengelola tag dengan menggunakan AWS X-Ray konsol atau API.

Anda dapat menerapkan tag ke sumber daya dengan menggunakan konsol X-Ray, API AWS CLI, SDKs, dan AWS Tools for Windows PowerShell. Untuk informasi selengkapnya, lihat dokumentasi berikut ini:

- API X-Ray – Lihat operasi berikut di Referensi API AWS X-Ray :
 - [ListTagsForResource](#)
 - [CreateSamplingRule](#)
 - [CreateGroup](#)
 - [TagResource](#)
 - [UntagResource](#)
- AWS CLI — Lihat [xray di Referensi AWS CLI](#) Perintah
- SDKs — Lihat dokumentasi SDK yang berlaku di halaman [AWS Dokumentasi](#)

Note

Jika Anda tidak dapat menambahkan atau mengubah tanda pada sumber daya X-Ray, atau Anda tidak dapat menambahkan sumber daya yang memiliki tanda tertentu, Anda

mungkin tidak memiliki izin untuk melakukan operasi ini. Untuk meminta akses, hubungi AWS pengguna di perusahaan Anda yang memiliki izin Administrator di X-Ray.

Topik

- [Batasan tag](#)
- [Mengelola tanda di konsol tersebut](#)
- [Mengelola tag di AWS CLI](#)
- [Kontrol akses ke sumber daya X-Ray berdasarkan tanda](#)

Batasan tag

Batasan berikut berlaku untuk tanda.

- Jumlah maksimum tanda per sumber daya – 50
- Panjang kunci maksimum – 128 karakter Unicode
- Panjang nilai maksimum – 256 karakter Unicode
- Nilai yang valid untuk kunci dan nilai – a-z, A-Z, 0-9, spasi, dan karakter berikut: `_ . : / = + -` dan `@`
- Kunci dan nilai tanda peka huruf besar-kecil.
- Jangan gunakan `aws :` sebagai prefiks untuk kunci; ini disimpan untuk penggunaan AWS .

Note

Anda tidak dapat mengedit atau menghapus tanda sistem.

Mengelola tanda di konsol tersebut

Anda dapat menambahkan tanda opsional saat membuat grup X-Ray atau aturan pengambilan sampel. Tanda juga dapat diubah atau dihapus di konsol tersebut nanti.

Prosedur berikut menjelaskan cara menambahkan, mengedit, dan menghapus tanda untuk grup dan aturan pengambilan sampel Anda di konsol X-Ray.

Topik

- [Tambahkan tanda ke grup baru \(konsol\)](#)
- [Tambahkan tanda ke aturan pengambilan sampel baru \(konsol\)](#)
- [Edit atau hapus tanda untuk grup \(konsol\)](#)
- [Edit atau hapus tanda untuk aturan pengambilan sampel \(konsol\)](#)

Tambahkan tanda ke grup baru (konsol)

Saat membuat grup X-Ray baru, Anda dapat menambahkan tanda opsional di halaman Buat grup.

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Di panel navigasi, luaskan Konfigurasi, dan pilih Grup.
3. Pilih Buat grup.
4. Pada halaman Buat grup, tentukan nama dan ekspresi filter untuk grup. Untuk informasi selengkapnya tentang properti ini, lihat [Mengkonfigurasi grup](#).
5. Di Tanda, masukkan kunci tanda dan, secara opsional, nilai tanda. Misalnya, Anda dapat memasukkan kunci tanda **Stage**, dan nilai tanda **Production**, untuk menunjukkan bahwa grup ini digunakan untuk produksi. Saat Anda menambahkan tanda, baris baru akan muncul bagi Anda untuk menambahkan tanda lain, jika diperlukan. Lihat [Batasan tag](#) di topik ini untuk keterbatasan pada tanda.
6. Setelah Anda selesai menambahkan tanda, pilih Buat grup.

Tambahkan tanda ke aturan pengambilan sampel baru (konsol)

Saat membuat aturan pengambilan sampel X-Ray baru, Anda dapat menambahkan tanda di halaman Buat aturan pengambilan sampel.

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Di panel navigasi, luaskan Konfigurasi, dan pilih Pengambilan Sampel.
3. Pilih Buat aturan pengambilan sampel.
4. Pada Buat aturan pengambilan sampel, tentukan nama, prioritas, batas, kriteria pencocokan, dan atribut yang cocok. Untuk informasi selengkapnya tentang properti ini, lihat [Mengonfigurasi aturan pengambilan sampel](#).

5. Di Tanda, masukkan kunci tanda dan, secara opsional, nilai tanda. Misalnya, Anda dapat memasukkan kunci tanda **Stage**, dan nilai tanda **Production**, untuk menunjukkan bahwa aturan pengambilan sampel ini digunakan untuk produksi. Saat Anda menambahkan tanda, baris baru akan muncul untuk menambahkan tanda lain, jika diperlukan. Lihat [Batasan tag](#) di topik ini untuk keterbatasan pada tanda.
6. Setelah Anda selesai menambahkan tanda, pilih Buat aturan pengambilan sampel.

Edit atau hapus tanda untuk grup (konsol)

Anda dapat mengubah atau menghapus tanda pada grup X-Ray di halaman Edit grup.

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Di panel navigasi, luaskan Konfigurasi, dan pilih Grup.
3. Di tabel Grup, pilih nama grup.
4. Pada halaman Edit grup, di Tanda, edit kunci dan nilai tanda. Anda tidak dapat memiliki kunci tanda duplikat. Nilai tanda bersifat opsional; Anda dapat menghapus nilai jika diinginkan. Untuk informasi selengkapnya tentang properti lain di halaman Edit grup, lihat [Mengonfigurasi grup](#). Lihat [Batasan tag](#) di topik ini untuk keterbatasan pada tanda.
5. Untuk menghapus tanda, pilih X di sebelah kanan tanda.
6. Setelah selesai mengedit atau menghapus tanda, pilih Perbarui grup.

Edit atau hapus tanda untuk aturan pengambilan sampel (konsol)

Anda dapat mengubah atau menghapus tanda pada aturan pengambilan sampel X-Ray di halaman Edit aturan pengambilan sampel.

1. Masuk ke AWS Management Console dan buka konsol X-Ray di <https://console.aws.amazon.com/xray/rumah>.
2. Di panel navigasi, luaskan Konfigurasi, dan pilih Pengambilan Sampel.
3. Di tabel Aturan pengambilan sampel, pilih nama aturan pengambilan sampel.
4. Di Tanda, edit kunci dan nilai tanda. Anda tidak dapat memiliki kunci tanda duplikat. Nilai tanda bersifat opsional; Anda dapat menghapus nilai jika diinginkan. Untuk informasi selengkapnya tentang properti lain di halaman Edit aturan pengambilan sampel, lihat [Mengonfigurasi aturan pengambilan sampel](#). Lihat [Batasan tag](#) di topik ini untuk keterbatasan pada tanda.

5. Untuk menghapus tanda, pilih X di sebelah kanan tanda.
6. Setelah selesai mengedit atau menghapus tanda, pilih Perbarui aturan pengambilan sampel.

Mengelola tag di AWS CLI

Anda dapat menambahkan tanda saat membuat grup X-Ray atau aturan pengambilan sampel. Anda juga dapat menggunakan AWS CLI untuk membuat dan mengelola tag. Untuk memperbarui tag pada grup atau aturan pengambilan sampel yang ada, gunakan AWS X-Ray konsol, atau atau. [TagResource](#)[UntagResource](#) APIs

Topik

- [Tambahkan tanda ke grup X-Ray atau aturan pengambilan sampel baru \(CLI\)](#)
- [Tambahkan tanda ke sumber daya yang ada \(CLI\)](#)
- [Daftar tanda pada sumber daya \(CLI\)](#)
- [Hapus tanda pada sumber daya \(CLI\)](#)

Tambahkan tanda ke grup X-Ray atau aturan pengambilan sampel baru (CLI)

Untuk menambahkan tanda opsional saat Anda membuat grup X-Ray atau aturan pengambilan sampel baru, gunakan salah satu perintah berikut.

- Untuk menambahkan tag ke grup baru, jalankan perintah berikut, ganti *group_name* dengan nama grup Anda, *mydomain.com* dengan titik akhir layanan Anda, *key_name* dengan kunci tag, dan opsional, *value* dengan nilai tag. Untuk informasi selengkapnya tentang cara membuat grup, lihat [Grup](#).

```
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"}]
```

Berikut adalah contohnya.

```
aws xray create-group \  
  --group-name "AdminGroup" \  
  --tags [{"Key": "Environment", "Value": "Production"}]
```

```
--filter-expression "service(\"mydomain.com\") {fault OR error}" \
--tags [{"Key": "Stage","Value": "Prod"}, {"Key": "Department","Value": "QA"}]
```

- Untuk menambahkan tag ke aturan sampling baru, jalankan perintah berikut, ganti *key_name* dengan kunci tag, dan opsional, *value* dengan nilai tag. Perintah ini menentukan nilai dalam parameter `--sampling-rule` sebagai file JSON. Untuk informasi selengkapnya tentang cara membuat aturan pengambilan sampel, lihat [Aturan pengambilan sampel](#).

```
aws xray create-sampling-rule \
--cli-input-json file://file_name.json
```

Berikut ini adalah isi dari file JSON *file_name.json* yang ditentukan oleh `--cli-input-json` parameter.

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


Berikut adalah contoh perintah tersebut.

```
aws xray create-sampling-rule \  
  --cli-input-json file://9000-base-scorekeep.json
```

Berikut ini adalah isi dari contoh file `9000-base-scorekeep.json` yang ditentukan oleh parameter `--cli-input-json`.

```
{  
  "SamplingRule": {  
    "RuleName": "base-scorekeep",  
    "ResourceARN": "*",  
    "Priority": 9000,  
    "FixedRate": 0.1,  
    "ReservoirSize": 5,  
    "ServiceName": "Scorekeep",  
    "ServiceType": "*",  
    "Host": "*",  
    "HTTPMethod": "*",  
    "URLPath": "*",  
    "Version": 1  
  }  
  "Tags": [  
    {  
      "Key": "Stage",  
      "Value": "Prod"  
    },  
    {  
      "Key": "Department",  
      "Value": "QA"  
    }  
  ]  
}
```

Tambahkan tanda ke sumber daya yang ada (CLI)

Anda dapat menjalankan perintah `tag-resource` untuk menambahkan tanda ke grup X-Ray atau aturan pengambilan sampel yang ada. Metode ini mungkin lebih sederhana daripada menambahkan tanda dengan menjalankan `update-group` atau `update-sampling-rule`.

Untuk menambahkan tanda ke grup atau aturan pengambilan sampel, jalankan perintah berikut, ganti ARN dengan ARN sumber daya, dan tentukan kunci dan nilai opsional tanda yang ingin Anda tambahkan.

```
aws xray tag-resource \  
  --resource-arn "ARN" \  
  --tag-keys [{"Key": "key_name", "Value": "value"}, {"Key": "key_name", "Value": "value"}]
```

Berikut adalah contohnya.

```
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"}]
```

Daftar tanda pada sumber daya (CLI)

Anda dapat menjalankan perintah `list-tags-for-resource` untuk daftar tanda dari grup X-Ray atau aturan pengambilan sampel.

Untuk mendaftarkan tanda yang terkait dengan grup atau aturan pengambilan sampel, jalankan perintah berikut, ganti ARN dengan ARN sumber daya.

```
aws xray list-tags-for-resource \  
  --resource-arn "ARN"
```

Berikut adalah contohnya.

```
aws xray list-tags-for-resource \  
  --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup"
```

Hapus tanda pada sumber daya (CLI)

Anda dapat menjalankan perintah `untag-resource` untuk menghapus tanda dari grup X-Ray atau aturan pengambilan sampel.

Untuk menghapus tanda dari grup atau aturan pengambilan sampel, jalankan perintah berikut, ganti ARN dengan ARN sumber daya, dan tentukan kunci tanda yang ingin Anda hapus.

Anda hanya dapat menghapus seluruh tanda dengan perintah `untag-resource`. Untuk menghapus nilai tanda, gunakan konsol X-Ray, atau hapus tanda dan tambahkan tanda baru dengan kunci yang sama, tetapi nilai yang berbeda atau kosong.

```
aws xray untag-resource \  
  --resource-arn "ARN" \  
  --tag-keys ["key_name", "key_name"]
```

Berikut adalah contohnya.

```
aws xray untag-resource \  
  --resource-arn "arn:aws:xray:us-east-2:01234567890:group/group_name" \  
  --tag-keys ["Stage", "Department"]
```

Kontrol akses ke sumber daya X-Ray berdasarkan tanda

Anda dapat melampirkan tanda ke grup X-Ray atau aturan pengambilan sampel, atau meneruskan tanda dalam permintaan ke X-Ray. Untuk mengendalikan akses berdasarkan tanda, Anda dapat memberikan informasi tentang tanda di kebijakan [elemen syarat](#) menggunakan kunci syarat `xray:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk mempelajari selengkapnya tentang kunci kondisi ini, lihat [Mengontrol akses ke AWS sumber daya menggunakan tag sumber daya](#).

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat [Mengelola akses ke grup X-Ray dan aturan pengambilan sampel berdasarkan tanda](#).

Pemecahan masalah AWS X-Ray

Topik ini mencantumkan kesalahan umum dan masalah yang mungkin Anda temui saat menggunakan X-Ray API, konsol, atau SDKs. Jika Anda menemukan masalah yang tidak tercantum di sini, Anda dapat menggunakan tombol Umpan Balik di halaman ini untuk melaporkannya.

Bagian-bagian

- [Peta jejak X-Ray dan halaman detail jejak](#)
- [X-Ray SDK for Java](#)
- [X-Ray SDK untuk Node.js](#)
- [X-Ray Daemon](#)

Peta jejak X-Ray dan halaman detail jejak

Bagian berikut dapat membantu jika Anda mengalami masalah menggunakan peta jejak X-Ray dan halaman detail Jejak:

Saya tidak melihat semua CloudWatch log saya

Cara mengkonfigurasi log sehingga muncul di peta jejak X-Ray dan halaman detail jejak tergantung pada layanan.

- Log API Gateway muncul jika log diaktifkan di API Gateway.

Tidak semua node peta layanan mendukung tampilan log terkait. Lihat log untuk jenis simpul berikut:

- Konteks Lambda
- Fungsi Lambda
- Tahap API Gateway
- Kluster Amazon ECS
- Contoh Amazon ECS
- Layanan Amazon ECS
- Tugas Amazon ECS
- Klaster Amazon EKS

- Ruang nama Amazon EKS
- Simpul Amazon EKS
- Pod Amazon EKS
- Layanan Amazon EKS

Saya tidak melihat semua alarm saya di peta jejak X-Ray

Peta jejak X-Ray hanya menampilkan ikon peringatan untuk node jika ada alarm yang terkait dengan node tersebut dalam status ALARM.

Peta jejak mengaitkan alarm dengan node menggunakan logika berikut:

- Jika node mewakili AWS layanan, maka semua alarm dengan namespace yang terkait dengan layanan tersebut terkait dengan node. Misalnya, node tipe `AWS::Kinesis` ditautkan dengan semua alarm yang didasarkan pada metrik di namespace `CloudWatch AWS/Kinesis`
- Jika node mewakili AWS sumber daya, maka alarm pada sumber daya tertentu ditautkan. Misalnya, simpul tipe `AWS::DynamoDB::Table` dengan nama "MyTable" ditautkan ke semua alarm yang didasarkan pada metrik dengan namespace `AWS/DynamoDB` dan `TableName` dimensi disetel ke `MyTable`
- Jika simpul memiliki jenis yang tidak diketahui, yang diidentifikasi dengan garis batas di sekitar nama, maka tidak ada alarm yang terkait dengan simpul tersebut.

Saya tidak melihat beberapa AWS sumber daya di peta jejak

Tidak semua AWS sumber daya diwakili oleh node khusus. Beberapa AWS layanan diwakili oleh satu node untuk semua permintaan ke layanan. Jenis sumber daya berikut ditampilkan dengan simpul per sumber daya:

- `AWS::DynamoDB::Table`
- `AWS::Lambda::Function`

Fungsi Lambda diwakili oleh dua node — satu untuk wadah Lambda, dan satu untuk fungsi tersebut. Fungsi ini membantu mengidentifikasi masalah awal yang dingin pada fungsi Lambda. Simpul wadah Lambda terkait dengan alarm dan dasbor dengan cara yang sama seperti simpul fungsi Lambda.

- `AWS::ApiGateway::Stage`

- `AWS::SQS::Queue`
- `AWS::SNS::Topic`

Ada terlalu banyak node di peta jejak

Gunakan grup Sinar-X untuk memecah peta Anda menjadi beberapa peta. Untuk informasi selengkapnya, lihat [Menggunakan Pernyataan Penyaring dengan Grup](#).

X-Ray SDK for Java

Kesalahan: Pengecualian di utas "Thread-1" `com.amazonaws.xray.exceptions`.

`SegmentNotFoundException`: Gagal memulai subsegmen bernama 'AmazonSNS': segmen tidak dapat ditemukan.

Kesalahan ini menunjukkan bahwa X-Ray SDK mencoba merekam panggilan keluar ke AWS, tetapi tidak dapat menemukan segmen terbuka. Hal ini dapat terjadi dalam situasi berikut:

- Sebuah filter servlet tidak dikonfigurasi – X-Ray SDK membuat segmen untuk permintaan masuk dengan filter bernama `AWSXRayServletFilter`. [Mengonfigurasi filter servlet](#) untuk instrumen permintaan masuk.
- Anda menggunakan klien terinstrumentasi di luar kode servlet – Jika Anda menggunakan klien terinstrumentasi untuk membuat panggilan dalam kode startup atau kode lain yang tidak berjalan dalam menanggapi permintaan masuk, Anda harus membuat segmen secara manual. Lihat [Menginstrumentasi kode perusahaan rintisan](#) sebagai contoh.
- Anda menggunakan klien terinstrumentasi dalam utas pekerja – Saat Anda membuat utas baru, pencatat X-Ray kehilangan referensi ke segmen terbuka. Anda dapat menggunakan metode [getTraceEntity](#) dan [setTraceEntity](#) untuk mendapatkan referensi ke segmen saat ini atau subsegmen ([Entity](#)), dan meneruskannya kembali ke pencatat di dalam utas. Lihat [Menggunakan klien berinstrumen di utas pekerja](#) sebagai contoh.

X-Ray SDK untuk Node.js

Masalah: CLS tidak berfungsi dengan Sequelize

Berikan X-Ray SDK untuk Node.js namespace untuk Sequelize dengan metode `cls`.

```
var AWSXRay = require('aws-xray-sdk');
```

```
const Sequelize = require('sequelize');  
Sequelize.cls = AWSXRay.getNamespace();  
const sequelize = new Sequelize(...);
```

Masalah: CLS tidak berfungsi dengan Bluebird

Gunakan `cls-bluebird` untuk membuat Bluebird berfungsi dengan CLS.

```
var AWSXRay = require('aws-xray-sdk');  
var Promise = require('bluebird');  
var clsBluebird = require('cls-bluebird');  
clsBluebird(AWSXRay.getNamespace());
```

X-Ray Daemon

Masalah: Daemon menggunakan kredensial yang salah

Daemon menggunakan AWS SDK untuk memuat kredensial. Jika Anda menggunakan beberapa metode untuk menyediakan kredensial, metode dengan prioritas tertinggi digunakan. Lihat [Menjalankan daemon](#) untuk informasi selengkapnya.

Riwayat Dokumen untuk AWS X-Ray

Tabel berikut menjelaskan perubahan penting pada dokumentasi untuk AWS X-Ray. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

Pembaruan dokumentasi terbaru: 07 Maret 2024

Perubahan	Deskripsi	Tanggal
Ditambahkan fungsionalitas	Migrasi dari X-Ray ke OpenTelemetry. Untuk informasi lebih lanjut, lihat Migrasi dari instrumentasi X-Ray ke OpenTelemetry instrumentasi .	Juni 13, 2025
Ditambahkan fungsionalitas	AWS X-Ray sekarang mendukung Pencarian Transaksi. Untuk informasi selengkapnya, lihat Pencarian Transaksi .	November 21, 2024
Ditambahkan fungsionalitas	AWS X-Ray sekarang mendukung OpenTelemetry Protocol (OTLP) Endpoint. Untuk informasi selengkapnya, lihat OpenTelemetry .	November 21, 2024
Ditambahkan fungsionalitas	X-Ray sekarang mencatat peristiwa data, termasuk PutTraceSegments, GetTraceSummaries, dan BatchGetTraces ke AWS CloudTrail. X-Ray juga sekarang mencatat acara GetSamplingStatisticSummaries.	7 Maret 2024

es manajemen ke CloudTrail. Untuk informasi selengkapnya, lihat [Logging panggilan X-Ray API dengan AWS CloudTrail](#).

[Ditambahkan fungsionalitas](#)

X-Ray sekarang mendukung jejak yang IDs dibuat melalui OpenTelemetry atau kerangka kerja lain yang sesuai dengan spesifikasi [W3C Trace Context](#). Untuk informasi selengkapnya, lihat [Mengirim data jejak ke X-Ray](#).

25 Oktober 2023

[Ditambahkan fungsionalitas](#)

Anda sekarang dapat mengonfigurasi penelusuran aktif Amazon SNS, memungkinkan Anda melacak dan menganalisis permintaan saat mereka melakukan perjalanan melalui topik Amazon SNS Anda. Untuk informasi selengkapnya, lihat [Amazon SNS](#) dan [AWS X-Ray](#).

Februari 8, 2023

[Diperbarui X-Ray SDK untuk topik Node.js](#)

Menambahkan detail untuk menginstrumentasi klien menggunakan AWS SDK untuk JavaScript V3. Untuk detailnya, lihat [Menelusuri panggilan AWS SDK dengan X-Ray SDK](#) untuk Node.js.

7 Februari 2023

[Detail kebijakan terkelola IAM yang diperbarui](#)

Menambahkan izin IAM untuk pengamatan lintas akun ke `AWSXRayReadOnlyAccess`, `AWSXRayFullAccess` dan `AWSXrayCrossAccountSharingConfiguration` kebijakan terkelola. Untuk detailnya, lihat [kebijakan terkelola IAM untuk X-Ray](#).

7 Februari 2023

[Ditambahkan fungsionalitas](#)

AWS X-Ray sekarang mendukung pengamatan lintas akun, memungkinkan Anda untuk memantau dan memecahkan masalah aplikasi yang menjangkau beberapa akun dalam file. Wilayah AWS Untuk detailnya, lihat [Penelusuran lintas akun](#).

27 November 2022

[Ditambahkan fungsionalitas](#)

Sekarang Anda dapat melihat jejak tertaut antara produsen pesan, antrean Amazon SQS, dan konsumen, memberikan tampilan tersambung dari jejak yang dikirim dari aplikasi berbasis peristiwa. Untuk informasi selengkapnya, lihat [melacak aplikasi berbasis peristiwa](#).

November 20, 2022

Detail kebijakan terkelola IAM yang diperbarui	Menambahkan izin IAM untuk mencantumkan kebijakan sumber daya ke kebijakan <code>AWSXRayReadOnlyAccess</code> terkelola. Untuk detailnya, lihat kebijakan terkelola IAM untuk X-Ray .	15 November 2022
Izin konsol IAM yang diperbarui dan detail kebijakan terkelola	Kumpulan izin IAM yang digunakan konsol X-Ray telah diperbarui, bersama dengan deskripsi kebijakan <code>AWSXRayReadOnlyAccess</code> terkelola. Untuk detailnya, lihat Menggunakan konsol X-Ray .	11 November 2022
Ditambahkan AWS Distro untuk Ruby OpenTelemetry	AWS Distro for OpenTelemetry (ADOT) menyediakan satu set open source APIs, perpustakaan, dan agen untuk mengumpulkan jejak dan metrik terdistribusi. ADOT Ruby memungkinkan Anda untuk instrumen aplikasi Ruby Anda untuk X-Ray dan back-end tracing lainnya. Untuk informasi lebih lanjut, lihat AWS Distro untuk OpenTelemetry Ruby .	Februari 7, 2022
Ditambahkan fungsionalitas	Anda sekarang dapat melihat jejak dan mengkonfigurasi X-Ray dari CloudWatch konsol. Untuk informasi selengkapnya, lihat Konsol X-Ray .	24 Januari 2022

[CloudWatch RUM Terpadu](#)

Dengan AWS X-Ray dan CloudWatch RUM, Anda dapat menganalisis dan men-debug jalur permintaan mulai dari pengguna akhir aplikasi Anda melalui layanan AWS terkelola hilir. Untuk informasi lebih lanjut, lihat [CloudWatch RUM dan AWS X-Ray](#).

Desember 3, 2021

[AWS Distro Terpadu untuk OpenTelemetry](#)

AWS Distro for OpenTelemetry (ADOT) menyediakan satu set open source APIs, perpustakaan, dan agen untuk mengumpulkan jejak dan metrik terdistribusi. ADOT memungkinkan Anda untuk instrumen aplikasi Anda untuk X-Ray dan back-end tracing lainnya. Untuk informasi selengkapnya, lihat [Menginstrumentasi aplikasi Anda](#).

September 23, 2021

[Ditambahkan fungsionalitas](#)

AWS X-Ray sekarang terintegrasi dengan Amazon Virtual Private Cloud, memungkinkan sumber daya di VPC Amazon Anda untuk berkomunikasi dengan layanan X-Ray tanpa melalui internet publik. Untuk informasi selengkapnya, lihat [Menggunakan AWS X-Ray dengan titik akhir VPC](#).

20 Mei 2021

Ditambahkan fungsionalitas	AWS X-Ray sekarang terintegrasi dengan AWS CloudFormation, memungkinkan Anda untuk menyediakan dan mengkonfigurasi sumber daya X-Ray. Untuk informasi selengkapnya, lihat Membuat sumber daya X-Ray dengan CloudFormation .	6 Mei 2021
Ditambahkan fungsionalitas	AWS X-Ray sekarang terintegrasi dengan Amazon EventBridge untuk melacak peristiwa yang dilewati EventBridge. Tindakan ini memberikan pandangan yang lebih lengkap kepada para pengguna tentang sistem mereka. Untuk informasi selengkapnya, lihat Amazon EventBridge dan AWS X-Ray .	2 Maret 2021
Menambahkan daemon ke ECR	Daemon sekarang dapat diunduh dari Amazon ECR. Untuk informasi selengkapnya, lihat Mengunduh daemon .	1 Maret 2021
Ditambahkan fungsionalitas	AWS X-Ray sekarang mendukung pemberitahuan terkait wawasan ke Amazon EventBridge. Ini memungkinkan Anda untuk mengambil tindakan otomatis pada wawasan menggunakan EventBridge. Untuk informasi selengkapnya, lihat Notifikasi Wawasan .	15 Oktober 2020

Menambahkan Daemon yang Dapat Diunduh

AWS X-Ray memperkenalkan daemon dukungan untuk Linux. ARM64 Untuk informasi selengkapnya, lihat [daemonbrazil ws AWS X-Ray](#)

1 Oktober 2020

Ditambahkan fungsionalitas

AWS X-Ray sekarang mendukung integrasi aktif dengan Amazon CloudWatch Synthetics. Hal ini memungkinkan Anda untuk melihat detail tentang simpul klien canary Synthetics seperti waktu dan status respons. Anda juga dapat melakukan analisis di konsol Analitik berdasarkan informasi dari simpul klien canary Synthetics. Untuk informasi lebih lanjut, lihat [Debugging kenari CloudWatch sintetis menggunakan](#) X-Ray.

24 September 2020

Ditambahkan fungsionalitas

AWS X-Ray sekarang mendukung penelusuran end-to-end alur kerja untuk AWS Step Functions. Anda dapat memvisualisasikan komponen status mesin, mengidentifikasi hambatan performa, dan memecahkan masalah permintaan yang mengakibatkan kesalahan. Untuk informasi lebih lanjut, lihat [AWS Step Functions dan AWS X-Ray](#).

14 September 2020

Ditambahkan fungsionalitas

AWS X-Ray memperkenalkan wawasan untuk terus menganalisis data jejak di akun Anda untuk mengidentifikasi masalah yang muncul dalam aplikasi Anda. Wawasan mencatat insiden dan melacak dampak insiden hingga resolusi. Untuk informasi selengkapnya, lihat [Menggunakan wawasan di konsol AWS X-Ray](#)

3 September 2020

Ditambahkan fungsionalitas

AWS X-Ray memperkenalkan agen instrumentasi otomatis Java, memungkinkan pelanggan untuk mengumpulkan data jejak tanpa harus memodifikasi aplikasi berbasis Java yang ada. Anda sekarang dapat melacak aplikasi berbasis web dan servlet Java dengan perubahan konfigurasi minimal dan tanpa perubahan kode. Untuk informasi selengkapnya, lihat [agen instrumentasi otomatis AWS X-Ray untuk Java](#).

3 September 2020

Ditambahkan fungsionalitas

AWS X-Ray telah menambahkan halaman Grup baru ke konsol X-Ray untuk membantu memudahkan pembuatan dan pengelolaan kelompok jejak. Untuk informasi selengkapnya, lihat [Mengonfigurasi grup di konsol X-Ray](#).

24 Agustus 2020

Ditambahkan fungsionalitas

AWS X-Ray sekarang memungkinkan Anda menambahkan tag ke grup dan aturan pengambilan sampel. Anda juga dapat mengontrol akses ke grup dan aturan pengambilan sampel berdasarkan tanda. Untuk informasi selengkapnya, lihat [Menandai aturan dan grup pengambilan sampel X-Ray](#) dan [Mengelola akses ke grup X-Ray dan aturan pengambilan sampel berdasarkan tanda](#).

24 Agustus 2020

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.